

Introduction

1.1 Overview

This specification defines the shell command language and the utilities provided by the X/Open System Interface (XSI). The utilities are accessed using commands given to command interpreters supporting the shell command language. The system interfaces and headers (described in CAE Specification, **System Interfaces and Headers, Issue 5**) and the utilities are jointly known as services to application programs. No particular restrictions are imposed on the way in which the services are implemented.

The utilities are defined in terms of their interface as seen from the *sh* command interpreter. Alternative interfaces are available to application programs through one of the *exec* functions, and the *popen()* and *system()* interfaces, all of which are described in CAE Specification, **System Interfaces and Headers, Issue 5**.

1.2 Conformance

An implementation conforming to this specification shall meet the following criteria:

- The system shall provide all the utilities described in this specification with all the functionality defined, subject to the following:
 - Development utilities listed in Section 1.3.1 on page 4 need not be provided.
 - FORTRAN utilities listed in Section 1.3.2 on page 4 need not be provided.
 - Legacy utilities listed in Section 1.3.3 on page 4 need not be provided.
 - Within the utilities that are provided, functionality marked by the codes OF, OP, PI, or UN (see Section 1.7 on page 8) need not be provided.
- The system may provide one or more of the following:
 - development utilities (as a group) listed in Section 1.3.1 on page 4
 - the FORTRAN77 compiler listed in Section 1.3.2 on page 4.

When an implementation claims that an optional or possibly unsupportable facility is provided, all of its constituent parts shall be provided and shall comply with the specification.

When an implementation claims that a development system is provided, all of the utilities marked **DEVELOPMENT** and listed in Section 1.3.1 on page 4, except *dis*, shall be provided and shall comply with the specification.

When an implementation claims that a FORTRAN system is provided, the utility marked **FORTRAN** and listed in Section 1.3.2 on page 4 shall be provided and shall comply with the specification.

Note: To determine whether an implementation supports development, FORTRAN or possibly unsupportable utilities, refer to the implementation's Conformance Statement.

- The system may provide additional or enhanced utilities and facilities not required by this specification, provided that such additions or enhancements do not affect the behaviour of an application that requires only the facilities described in this specification.

An implementation conforming to this specification depends on the environment provided by the system interfaces and headers specified in the referenced **XSH** specification. For further information, refer to the **XSH** specification, **Section 1.2, Conformance**, the referenced **XPG4, Version 2** document and the implementation's Conformance Statement.

1.2.1 Symbolic Links

EX The definition of symbolic links in the **XBD** specification, **Chapter 2, Glossary** was new in Issue 4, Version 2. An implementation need not support symbolic links to be conformant with this document. (See the **XSH** specification, **Section 1.2, Conformance** for the implications on conformance with the **XSH** specification.)

The definition of pathname resolution in the **XBD** specification, **Chapter 2, Glossary** indicates the effects of symbolic links. However, many utilities that manipulate files may manipulate symbolic links. Use of these utilities in this context (that is, when the last component of the pathname is a symbolic link) produces unspecified effects. In addition, if any utility encounters a symbolic link after pathname resolution has been completed, the effects are unspecified.

For future directions, see Section 1.6.1 on page 7.

1.2.2 Considerations for Utilities in Support of Files of Arbitrary Size

EX The following utilities will support files of any size up to the maximum that can be created by the implementation. This support includes correct writing of file size-related values (such as file sizes and offsets, line numbers, and block counts) and correct interpretation of command line arguments that contain such values.

<i>basename</i>	Return non-directory portion of pathname.
<i>cat</i>	Concatenate and print files.
<i>cd</i>	Change working directory.
<i>chgrp</i>	Change file group ownership.
<i>chmod</i>	Change file modes.
<i>chown</i>	Change file ownership.
<i>cksum</i>	Write file checksums and sizes.
<i>cmp</i>	Compare two files.
<i>cp</i>	Copy files.
<i>dd</i>	Convert and copy a file.
<i>df</i>	Report free disk space.
<i>dirname</i>	Return directory portion of pathname.
<i>du</i>	Estimate file space usage.
<i>find</i>	Find files.
<i>ln</i>	Link files.
<i>ls</i>	List directory contents.
<i>mkdir</i>	Make directories.
<i>mv</i>	Move files.
<i>pathchk</i>	Check pathnames.
<i>pwd</i>	Return working directory name.
<i>rm</i>	Remove directory entries.
<i>rmdir</i>	Remove directories.
<i>sh</i>	Shell, the standard command language interpreter.
<i>sum</i>	Print checksum and block or byte count of a file.

84	<i>test</i>	Evaluate expression.	
85	<i>touch</i>	Change file access and modification times.	
86	<i>ulimit</i>	Set or report file size limit.	
87	Exceptions to the requirement that utilities support files of any size up to the maximum are:		
88	1. Utilities such as <i>tar</i> and <i>cpio</i> cannot support arbitrary file sizes due to limitations imposed		
89	by fixed file formats.		
90	2. Uses of files as command scripts, or for configuration or control, are exempt. For example,		
91	it is not required that <i>sh</i> be able to read an arbitrarily large “ <i>.profile</i> ”.		
92	3. Shell input and output redirection are exempt. For example, it is not required that the		
93	redirections <i>sum < file</i> or <i>echo foo > file</i> succeed for an arbitrarily large existing file.		
94			

1.3 Options

Utilities marked **DEVELOPMENT**, **FORTTRAN** or **LEGACY** are optional. See Section 1.2 on page 1 for conformance requirements.

1.3.1 Development

Utilities marked **DEVELOPMENT** in this specification are:

Development Utilities			
<i>admin</i>	<i>get</i>	<i>nm</i>	<i>strip</i>
<i>cflow</i>	<i>lex</i>	<i>prs</i>	<i>unget</i>
<i>ctags</i>	<i>lint</i>	<i>rmdel</i>	<i>val</i>
<i>cxref</i>	<i>m4</i>	<i>sact</i>	<i>what</i>
<i>delta</i>	<i>make</i>	<i>sccs</i>	<i>yacc</i>
<i>dis*</i>			

1.3.2 FORTRAN

The *fort77* FORTRAN compiler is the only utility marked **FORTTRAN** in this specification.

1.3.3 LEGACY

The utilities in the table below are marked **LEGACY**. Various factors may have contributed to the decision to class a utility **LEGACY**. Application writers should not use functionality marked **LEGACY**.

If a migration path exists, advice is given to application developers regarding alternative means of obtaining similar functionality. This information may be found in the APPLICATION USAGE sections on the relevant pages.

No requirement beyond that which was in effect at the time that these utilities were marked **LEGACY** shall be applied to these utilities.

Legacy Utilities			
<i>calendar</i>	<i>dis*</i>	<i>mail</i>	<i>unpack</i>
<i>cancel</i>	<i>du</i>	<i>pack</i>	<i>uulog</i>
<i>cc</i>	<i>egrep</i>	<i>pcat</i>	<i>uuname</i>
<i>col</i>	<i>fgrep</i>	<i>pg</i>	<i>uupick</i>
<i>cpio</i>	<i>line</i>	<i>spell</i>	<i>uuto</i>
<i>cu</i>	<i>lint</i>	<i>sum</i>	
<i>dircmp</i>	<i>lpstat</i>	<i>tar</i>	

* Even if the **DEVELOPMENT** option is supported, or the **LEGACY** option is supported, the *dis* utility need not be supported.

1.4 Changes from Issue 4

The following sections describe changes made to this specification since Issue 4. The CHANGE HISTORY section for each utility describes technical changes made to that utility since Issue 4. Changes made between Issue 2 and Issue 4 are not included.

1.4.1 Changes from Issue 4 to Issue 4, Version 2

Addition of the X/Open UNIX extension which specifies the common core utilities of 4.3 Berkeley Software Distribution (BSD 4.3), the OSF AES and SVID Issue 3.

1.4.2 Changes from Issue 4, Version 2 to Issue 5

- Addition of the Large File Summit extensions.
- Update of some utilities to reflect changes for the POSIX Realtime Extension.
- Update of some utilities to reflect changes for the POSIX Threads Extension.
- Introduction of the LEGACY category of utilities as a replacement for the TO BE WITHDRAWN, WITHDRAWN and Possibly Unsupportable categories.
- Addition of the following additional utilities:

New Utilities	
<i>fuser</i>	<i>link</i>
<i>ipcrm</i>	<i>unlink</i>
<i>ipcs</i>	

1.5 Terminology

The following terms are used in this specification:

can

This describes a permissible optional feature or behaviour available to the user or application; all systems support such features or behaviour as mandatory requirements.

implementation-dependent

The value or behaviour is not consistent across all implementations. The provider of an implementation normally documents the requirements for correct program construction and correct data in the use of that value or behaviour. When the value or behaviour in the implementation is designed to be variable or customisable on each instantiation of the system, the provider of the implementation normally documents the nature and permissible ranges of this variation. Applications that are intended to be portable must not rely on implementation-dependent values or behaviour.

legacy

Certain features are *legacy*, which means that they are being retained for compatibility with older applications, but have limitations which make them inappropriate for developing portable applications. New applications should use alternative means of obtaining equivalent functionality. Legacy features are marked **LEGACY**.

may

With respect to implementations, the feature or behaviour is optional. Applications should not rely on the existence of the feature. To avoid ambiguity, the reverse sense of *may* is expressed as *need not*, instead of *may not*.

must

This describes a requirement on the application or user.

should

With respect to implementations, the feature is recommended, but it is not mandatory. Applications should not rely on the existence of the feature.

With respect to users or applications, the word means recommended programming practice that is necessary for maximum portability.

undefined

A value or behaviour is undefined if this document imposes no portability requirements on applications for erroneous program constructs or erroneous data. Implementations may specify the result of using that value or causing that behaviour, but such specifications are not guaranteed to be consistent across all implementations. An application using such behaviour is not fully portable to all systems.

unspecified

A value or behaviour is unspecified if this document imposes no portability requirements on applications for correct program construct or correct data. Implementations may specify the result of using that value or causing that behaviour, but such specifications are not guaranteed to be consistent across all implementations. An application requiring a specific behaviour, rather than tolerating any behaviour when using that functionality, is not fully portable to all systems.

will

This means that the behaviour described is a requirement on the implementation and applications can rely on its existence.

1.6 Relationship to Formal Standards

Great care has been taken to ensure that this document is fully aligned with the following formal standards:

- ISO/IEC 9945-1: 1996
- ISO/IEC 9945-2: 1993
- ISO/IEC 9899: 1990
- ISO/IEC 9899:1990/Amendment 1:1994 (E) (MSE).

Any conflict between this document and any of these standards is unintentional. This document defers to the formal standards, which The Open Group recognises as superior. In particular, from time to time, when ambiguities are found in the formal standards, the responsible bodies will make interpretations of them, whose findings become binding on the standard. Where, as the result of such an interpretation, or for any other reason, any of these formal standards are found to conflict with this document, XSI-conformant systems are required to behave in the manner defined either by the formal standard or by this document. Application writers should clearly avoid depending exclusively on either behaviour in such cases; the list of all conflicts found since publication of this document is available on request. (See page ii for how to contact The Open Group.)

1.6.1 Relationship to Emerging Formal Standards

A future edition of this specification will be fully-aligned with the emerging IEEE 1003.1a and 1003.2b standards. This will fully specify the behaviour of utilities in reference to symbolic links.

1.7 Portability

Some of the utilities in this document and functions in CAE Specification, **System Interfaces and Headers, Issue 5** describe functionality that might not be fully portable to systems based on the ISO POSIX-1 or ISO POSIX-2 standards. Where enhanced or reduced functionality is specified, the text is shaded and a code in the margin identifies the nature of the extension or warning (see Section 1.7.1). For maximum portability, an application should avoid such functionality.

Unless the primary task of a utility is to produce textual material on its standard output, application developers should not rely on the format or content of any such material that may be produced. Where the primary task is to provide such material, but the output format is incompletely specified, the description is marked. Application developers are warned not to expect that the output of such an interface on one system will be any guide to its behaviour on another system.

1.7.1 Codes

The codes and their meanings are as follows:

EX Extension.

The functionality described is an extension to the standards referenced above. Application writers may confidently make use of an extension as it will be supported on all XSI-conformant systems. These extensions are designed not to conflict with the published standards.

If an entire **SYNOPSIS** section is shaded and marked with one EX, all the functionality described in that entry is an extension.

Some behaviour which is allowed to be optional in the formal standards is mandated on XSI-conformant systems. Such behaviours (for example, those dependent on the availability of job control) might not be individually marked as extensions, but the mandatory nature of the feature is marked as an extension where the option is described, typically in the header where the corresponding symbolic constant is defined.

FIPS FIPS Requirements.

The **Federal Information Processing Standards (FIPS)** are a series of U.S. government procurement standards managed and maintained on behalf of the U.S. Department of Commerce by the National Institute of Standards and Technology (NIST). Where restrictions have been made in order to align with the FIPS requirements, they have the special mark shown here, and appear in the index under FIPS alignment (as well as under EX).

The following restrictions are required by FIPS 151-2:

- The implementation will support `{_POSIX_CHOWN_RESTRICTED}`.
- The limit `{NGROUPS_MAX}` will be greater than or equal to 8.
- The implementation will support the setting of the group ID of a file (when it is created) to that of the parent directory.
- The implementation will support `{_POSIX_SAVED_IDS}`.
- The implementation will support `{_POSIX_VDISABLE}`.
- The implementation will support `{_POSIX_JOB_CONTROL}`.
- The implementation will support `{_POSIX_NO_TRUNC}`.
- The `read()` call returns the number of bytes read when interrupted by a signal and will not return `-1`.

256		• The <code>write()</code> call returns the number of bytes written when interrupted by a signal and will
257		not return <code>-1</code> .
258		• In the environment for the login shell, the environment variables <code>LOGNAME</code> and <code>HOME</code> will
259		be defined and have the properties described in Chapter 5 of CAE Specification, System
260		Interface Definitions, Issue 5 .
261		• The value of <code>{CHILD_MAX}</code> will be greater than or equal to 25.
262		• The value of <code>{OPEN_MAX}</code> will be greater than or equal to 20.
263		• The implementation will support the functionality associated with the symbols <code>CS7</code> , <code>CS8</code> ,
264		<code>CSTOPB</code> , <code>PARODD</code> and <code>PARENB</code> defined in <code><termios.h></code> .
265	JC	Job Control Extension.
266		Job control is an optional feature in the operating system described by the ISO POSIX-1
267		standard, but it is supported by all XSI-conformant systems. When interfaces rely on this
268		extension, they have the special mark shown here and appear in the index under JC (in addition
269		to being under EX).
270	OB	Obsolescent.
271		Some of the interfaces describe functionality that is obsolescent. Although these are fully
272		portable to all current XSI-conformant systems they may be withdrawn in future issues.
273	OF	Output format incompletely specified.
274		The format of the output produced by the utility is not fully specified. It is therefore not possible
275		to post-process this output in a consistent fashion. Typical problems include unknown length of
276		strings and unspecified field delimiters.
277	OP	Dependent on optional service in XSI.
278		Typical implementations depend on an optional service and the functionality affected need not
279		be present if the optional service is not supported.
280	PI	The behaviour cannot be guaranteed to be consistent.
281		It is not possible to guarantee that the interface behaves in the same way on all XSI-conformant
282		systems. This is the case if it provides functionality that is system-defined or system-specific.
283		Options that are used to <i>select</i> alternative forms of system-specific behaviour are not marked, as
284		it is clear from their descriptions that their use is inherently non-portable.
285	UN	Possibly unsupportable feature.
286		It need not be possible to implement the required functionality (as defined) on all XSI-
287		conformant systems and the functionality need not be present. This may, for example, be the
288		case where the XSI-conformant system is hosted and the underlying system provides the service
289		in an alternative way.

1.8 Grammar Conventions

Portions of this specification are expressed in terms of a special grammar notation. It is used to portray the complex syntax of certain program input. The grammar is based on the syntax used by the *yacc* utility. However, it does not represent fully functional *yacc* input, suitable for program use; the lexical processing and all semantic requirements are described only in textual form. The grammar is not based on source used in any traditional implementation and has not been tested with the semantic code that would normally be required to accompany it. Furthermore, there is no implication that the partial *yacc* code presented represents the most efficient, or only, means of supporting the complex syntax within the utility. Implementations may use other programming languages or algorithms, as long as the syntax supported is the same as that represented by the grammar.

The following typographical conventions are used in the grammar; they have no significance except to aid in reading.

- The identifiers for the reserved words of the language are shown with a leading capital letter. (These are terminals in the grammar. Examples: **While**, **Case**.)
- The identifiers for terminals in the grammar are all named with upper-case letters and underscores. Examples: **NEWLINE**, **ASSIGN_OP**, **NAME**.
- The identifiers for non-terminals are all lower-case.

1.9 Utility Description Defaults

This section describes all of the subsections used within the utility descriptions, including:

- intended usage of the section
- global defaults that affect all the standard utilities
- the meanings of notations used in the standard that are specific to individual utility sections.

Integer variables and constants, including the values of operands and option-arguments, used by the utilities listed in this specification shall be implemented as equivalent to the ISO C standard **signed long** data type. Conversion between types shall be as described in the ISO C standard. The evaluation of arithmetic expressions shall be equivalent to that described in Section 6.3 of the ISO C standard.

SYNOPSIS

The SYNOPSIS section summarises the syntax of the calling sequence for the utility, including options, option-arguments and operands. Standards for utility naming are described in **XBD** specification, **Section 10.2, Utility Syntax Guidelines**; for describing the utility's arguments in **XBD** specification, **Section 10.1, Utility Argument Syntax**.

DESCRIPTION

The DESCRIPTION section describes the actions of the utility. If the utility has a very complex set of subcommands or its own procedural language, an EXTENDED DESCRIPTION section is also provided. Most explanations of optional functionality are omitted here, as they are usually explained in the OPTIONS section.

Some utilities in this specification are described in terms of functionality equivalent to the **XSH** specification. When specific functions are cited, the underlying operating system provides equivalent functionality and all side effects associated with successful execution of the function. The treatment of errors and intermediate results from the individual functions cited are generally not specified by this specification. See the utility's EXIT STATUS and CONSEQUENCES OF ERRORS sections for all actions associated with errors encountered by the utility.

OPTIONS

The OPTIONS section describes the utility options and option-arguments, and how they modify the actions of the utility. Standard utilities that have options either fully comply with the **XBD** specification, **Section 10.2, Utility Syntax Guidelines** or describe all deviations. Apparent disagreements between functionality descriptions in the OPTIONS and DESCRIPTION (or EXTENDED DESCRIPTION) sections are always resolved in favour of the OPTIONS section.

Each OPTIONS section that uses the phrase “The ... utility supports the Utility Syntax Guidelines ...” refers only to the use of the utility as specified by this specification; implementation extensions should also conform to the guidelines, but may allow exceptions for historical practice.

Unless otherwise stated in the utility description, when given an option unrecognised by the implementation, or when a required option-argument is not provided, standard utilities will issue a diagnostic message to standard error and exit with a non-zero exit status.

EX All utilities in this specification are capable of processing arguments using 8-bit transparency.

Default Behaviour: When this section is listed as “None”, it means that the implementation need not support any options. Standard utilities that do not accept

options, but that do accept operands, will recognise `--` as a first argument to be discarded.

The requirement for recognising `--` is because portable applications need a way to shield their operands from any arbitrary options that the implementation may provide as an extension. For example, if the standard utility *foo* is listed as taking no options, and the application needed to give it a pathname with a leading hyphen, it could safely do it as:

```
foo -- -myfile
```

and avoid any problems with `-m` used as an extension.

OPERANDS

The OPERANDS section describes the utility operands, and how they affect the actions of the utility. Apparent disagreements between functionality descriptions in the OPERANDS and DESCRIPTION (or EXTENDED DESCRIPTION) sections are always resolved in favour of the OPERANDS section.

If an operand naming a file can be specified as `"-"`, which means to use the standard input instead of a named file, this is explicitly stated in this section. Unless otherwise stated, the use of multiple instances of `"-"` to mean standard input in a single command produces unspecified results.

Unless otherwise stated, the standard utilities that accept operands will process those operands in the order specified in the command line.

Default Behaviour: When this section is listed as "None", it means that the implementation need not support any operands.

STDIN

The STDIN section describes the standard input of the utility. This section is frequently merely a reference to the following section, as many utilities treat standard input and input files in the same manner. Unless otherwise stated, all restrictions described in the INPUT FILES section apply to this section as well.

Use of a terminal for standard input can cause any of the standard utilities that read standard input to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

The specified standard input format of the standard utilities does not depend on the existence or value of the environment variables defined in this specification, except as provided by this specification.

Default Behaviour: When this section is listed as "Not used," it means that the standard input will not be read when the utility is used as described by this specification.

INPUT FILES

The INPUT FILES section describes the files, other than the standard input, used as input by the utility. It includes files named as operands and option-arguments as well as other files that are referred to, such as startup and initialisation files, databases, and so on. Commonly-used files are generally described in one place and cross-referenced by other utilities.

All utilities in this specification are capable of processing input files using 8-bit transparency.

When a standard utility reads a seekable input file and terminates without an error before it reaches end-of-file, the utility will ensure that the file offset in the open file description is properly positioned just past the last byte processed by the utility. For files that are not seekable, the state of the file offset in the open file description for that file is unspecified. A portable application cannot assume that the following three commands are equivalent:

```
tail -n +2 file
(sed -n 1q; cat) < file
cat file | (sed -n 1q; cat)
```

The second command is equivalent to the first only when the file is seekable. The third command leaves the file offset in the open file description in an unspecified state. Other utilities, such as *head*, *read* and *sh*, have similar properties.

Some of the standard utilities, such as filters, process input files a line or a block at a time and have no restrictions on the maximum input file size. Some utilities may have size limitations that are not as obvious as file space or memory limitations. Such limitations should reflect resource limitations of some sort, not arbitrary limits set by implementors. Implementations will document those utilities that are limited by constraints other than file system space, available memory and other limits specifically cited by this specification, and identify what the constraint is and indicate a way of estimating when the constraint would be reached. Similarly, some utilities descend the directory tree (recursively). Implementations will also document any limits that they may have in descending the directory tree that are beyond limits cited by this specification.

When an input file is described as a *text file*, the utility produces undefined results if given input that is not from a text file, unless otherwise stated. Some utilities (for example, *make*, *read*, *sh*) allow for continued input lines using an escaped <newline> convention; unless otherwise stated, the utility need not be able to accumulate more than {LINE_MAX} bytes from a set of multiple, continued input lines. Thus, for a portable application the total of all the continued lines in a set cannot exceed {LINE_MAX}. If a utility using the escaped <newline> convention detects an end-of-file condition immediately after an escaped <newline>, the results are unspecified.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See **XBD specification, Chapter 3, File Format Notation** for a description of this notation. The format description is intended to be sufficiently rigorous to allow other applications to generate these input files. However, since <blank> characters can legitimately be included in some of the fields described by the standard utilities, particularly in locales other than the POSIX locale, this intent is not always realised.

Default Behaviour: When this section is listed as “None”, it means that no input files are required to be supplied when the utility is used as described by this specification.

ENVIRONMENT VARIABLES

The ENVIRONMENT VARIABLES section lists what variables affect the utility's execution.

The entire manner in which environment variables described in this specification affect the behaviour of each utility is described in the ENVIRONMENT VARIABLES section for that utility, in conjunction with the global effects of the *LANG*, *LC_ALL* and *NLSPATH* environment variables described in **XBD specification, Chapter 6, Environment Variables**. The existence or value of environment variables described in

this document will not otherwise affect the specified behaviour of the standard utilities. Any effects of the existence or value of environment variables not described by this specification upon the standard utilities are unspecified.

For those standard utilities that use environment variables as a means for selecting a utility to execute (such as *CC* in *make*), the string provided to the utility is subjected to the path search described for *PATH* in the **XBD** specification, **Chapter 6, Environment Variables**.

EX All utilities in this specification are capable of processing environment variable names and values using 8-bit transparency.

Default Behaviour: When this section is listed as “None”, it means that the behaviour of the utility is not directly affected by environment variables described by this specification when the utility is used as described by this specification.

ASYNCHRONOUS EVENTS

The ASYNCHRONOUS EVENTS section lists how the utility reacts to such events as signals and what signals are caught.

Default Behaviour: When this section is listed as “Default”, or it refers to “the standard action for all other signals; see Section 1.9 on page 11” it means that the action taken as a result of the signal is one of the following:

1. The action is that inherited from the parent according to the rules of inheritance of signal actions defined in the **XSH** specification.
2. When no action has been taken to change the default, the default action is that specified by the **XSH** specification.
3. The result of the utility’s execution is as if default actions had been taken.

A utility is permitted to catch a signal, perform some additional processing (such as deleting temporary files), restore the default signal action (or action inherited from the parent process) and resignal itself.

STDOUT

The STDOUT section describes the standard output of the utility. This section is frequently merely a reference to the following section, **OUTPUT FILES**, because many utilities treat standard output and output files in the same manner.

Use of a terminal for standard output may cause any of the standard utilities that write standard output to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See the **XBD** specification, **Chapter 3, File Format Notation** for a description of this notation.

The specified standard output of the standard utilities does not depend on the existence or value of the environment variables defined in this specification, except as provided by this specification.

Some of the standard utilities describe their output using the verb *display*, defined in the **XBD** specification, **Chapter 2, Glossary**. Output described in the STDOUT sections of such utilities may be produced using means other than standard output. When standard output is directed to a terminal, the output described will be written directly to the terminal. Otherwise, the results are undefined.

Default Behaviour: When this section is listed as “Not used”, it means that the standard output will not be written when the utility is used as described by this specification.

STDERR

The STDERR section describes the standard error output of the utility. Only those messages that are purposely sent by the utility are described.

Use of a terminal for standard error may cause any of the standard utilities that write standard error output to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

The format of diagnostic messages for most utilities is unspecified, but the language and cultural conventions of diagnostic and informative messages whose format is unspecified by this standard should be affected by the setting of *LC_MESSAGES* and *NLSPATH*.

The specified standard error output of standard utilities does not depend on the existence or value of the environment variables defined in this specification, except as provided by this specification.

Default Behaviour: When this section is listed as “Used only for diagnostic messages,” it means that, unless otherwise stated, the diagnostic messages are sent to the standard error only when the exit status is non-zero and the utility is used as described by this specification.

When this section is listed as “Not used”, it means that the standard error will not be used when the utility is used as described in this specification.

This section does not describe error messages that refer to incorrect operation of the utility. Consider a utility that processes program source code as its input. This section is used to describe messages produced by a correctly operating utility that encounters an error in the program source code on which it is processing. However, a message indicating that the utility had insufficient memory in which to operate would not be described.

Some compilers have traditionally produced warning messages without returning a non-zero exit status; these are specifically noted in their sections. Other utilities are expected to remain absolutely quiet on the standard error if they want to return zero, unless the implementation provides some sort of extension to increase the verbosity or debugging level.

OUTPUT FILES

The OUTPUT FILES section describes the files created or modified by the utility. Temporary or system files that are created for internal usage by this utility or other parts of the implementation (for example, spool, log and audit files) are not described in this, or any, section. The utilities creating such files and the names of such files are unspecified. If applications are written to use temporary or intermediate files, they should use the *TMPDIR* environment variable, if it is set and represents an accessible directory, to select the location of temporary files.

Temporary files used by the standard utilities are named so that different utilities or multiple instances of the same utility can operate simultaneously without regard to their working directories, or any other process characteristic other than process ID. There are two exceptions to this rule:

1. Resources for temporary files other than the name space (for example, disk space, available directory entries, or number of processes allowed) are not guaranteed.
2. Certain standard utilities generate output files that are intended as input for other utilities, (for example, *lex* generates *lex.yy.c*) and these cannot have unique names. These cases are explicitly identified in the descriptions of the respective utilities.

Any temporary file created by the implementation is removed by the implementation upon a utility's successful exit, exit because of errors, or before termination by any of the SIGHUP, SIGINT or SIGTERM signals, unless specified otherwise by the utility description.

Receipt of the SIGQUIT signal should generally cause termination (unless in some debugging mode) that would bypass any attempted recovery actions.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See the **XBD** specification, **Chapter 3, File Format Notation** for a description of this notation.

Default Behaviour: When this section is listed as "None", it means that no files are created or modified as a consequence of direct action on the part of the utility when the utility is used as described by this specification. However, the utility may create or modify system files, such as log files, that are outside the utility's normal execution environment.

EXTENDED DESCRIPTION

The EXTENDED DESCRIPTION section provides a place for describing the actions of very complicated utilities, such as text editors or language processors, which typically have elaborate command languages.

Default Behaviour: When this section is listed as "None", no further description is necessary.

EXIT STATUS

The EXIT STATUS section describes the values the utility will return to the calling program, or shell, and the conditions that cause these values to be returned. Usually, utilities return zero for successful completion and values greater than zero for various error conditions. If specific numeric values are listed in this section, the system will use those values for the errors described. In some cases, status values are listed more loosely, such as ">0". A portable application cannot rely on any specific value in the range shown and must be prepared to receive any value in the range.

For example, a utility may list zero as a successful return, 1 as a failure for a specific reason, and >1 as "an error occurred". In this case, unspecified conditions may cause a 2 or 3, or other value, to be returned. A portable application should be written so that it tests for successful exit status values (zero in this case), rather than relying upon the single specific error value listed in this specification. In that way, it will have maximum portability, even on implementations with extensions.

Unspecified error conditions may be represented by specific values not listed in this specification.

CONSEQUENCES OF ERRORS

The CONSEQUENCES OF ERRORS section describes the effects on the environment, file systems, process state, and so on, when error conditions occur. It does not describe error messages produced or exit status values used.

The many reasons for failure of a utility are generally not specified by the utility descriptions. Utilities may terminate prematurely if they encounter: invalid usage of options, arguments or environment variables; invalid usage of the complex syntaxes expressed in EXTENDED DESCRIPTION sections; difficulties accessing, creating, reading or writing files; or difficulties associated with the privileges of the process.

The following apply to each utility, unless otherwise stated:

- If the requested action cannot be performed on an operand representing a file, directory, user, process, and so on, the utility will issue a diagnostic message to standard error and continue processing the next operand in sequence, but the final exit status is returned as non-zero.

For a utility that recursively traverses a file hierarchy (such as *find* or *chown -R*), if the requested action cannot be performed on a file or directory encountered in the hierarchy, the utility will issue a diagnostic message to standard error and continue processing the remaining files in the hierarchy, but the final exit status will be returned as non-zero.

- If the requested action characterised by an option or option-argument cannot be performed, the utility will issue a diagnostic message to standard error and the exit status returned will be non-zero.
- When an unrecoverable error condition is encountered, the utility will exit with a non-zero exit status.
- A diagnostic message will be written to standard error whenever an error condition occurs.

When a utility encounters an error condition several actions are possible, depending on the severity of the error and the state of the utility. Included in the possible actions of various utilities are: deletion of temporary or intermediate work files; deletion of incomplete files; validity checking of the file system or directory.

Default Behaviour: When this section is listed as “Default”, it means that any changes to the environment are unspecified.

APPLICATION USAGE

The APPLICATION USAGE section gives advice to the application programmer or user about the way the utility should be used.

EXAMPLES

The EXAMPLES section gives one or more examples of usage, where appropriate. This section is non-normative. In the event of conflict between an example and a normative part of the specification, the normative material is to be taken as correct.

In all examples, quoting has been used, showing how sample commands (utility names combined with arguments) could be passed correctly to a shell (see *sh*) or as a string to the **XSH** specification *system()* function. Such quoting would not be used if the utility is invoked using one of the **XSH** specification *exec* functions.

FUTURE DIRECTIONS

The FUTURE DIRECTIONS section should be used as a guide to current thinking; there is not necessarily a commitment to implement all of these future directions in their entirety.

SEE ALSO

The SEE ALSO section lists related entries.

CHANGE HISTORY

The CHANGE HISTORY section shows the derivation of the description used by the XSI and lists the functional differences between Issues 4 and 5. Detailed listings of updates performed to align with the **XPG4, Version 2** document are not given.

Certain of the standard utilities describe how they can invoke other utilities or applications, such as by passing a command string to the command interpreter. The external influences (STDIN, ENVIRONMENT VARIABLES, and so on) and external effects (STDOUT, CONSEQUENCES OF ERRORS, and so on) of such invoked utilities are not described in the section concerning the standard utility that invokes them.

Shell Command Language

636

637

This chapter contains the definition of the XSI Shell Command Language.

638

2.1 Shell Introduction

639

640

641

The shell is a command language interpreter. This chapter describes the syntax of that command language as it is used by the *sh* utility and the *system()* and *popen()* functions in the **XSH** specification.

642

643

The shell operates according to the following general overview of operations. The specific details are included in the cited sections of this chapter.

644

645

646

1. The shell reads its input from a file (see *sh*), from the *-c* option or from the *system()* and *popen()* functions in the **XSH** specification. If the first line of a file of shell commands starts with the characters *#!*, the results are unspecified.

647

648

649

The construct *#!* is reserved for implementations wishing to provide that extension. A portable application cannot use *#!* as the first line of a shell script; it might not be interpreted as a comment.

650

2. The shell breaks the input into tokens: words and operators. (See Section 2.3 on page 23.)

651

652

3. The shell parses the input into simple commands (see Section 2.9.1 on page 45) and compound commands (see Section 2.9.4 on page 52).

653

654

655

4. The shell performs various expansions (separately) on different parts of each command, resulting in a list of pathnames and fields to be treated as a command and arguments (see Section 2.6 on page 31).

656

657

5. The shell performs redirection (see Section 2.7 on page 40) and removes redirection operators and their operands from the parameter list.

658

659

660

661

662

6. The shell executes a function (see Section 2.9.5 on page 54), built-in (see Section 2.14 on page 67), executable file or script, giving the names of the arguments as positional parameters numbered 1 to *n*, and the name of the command (or in the case of a function within a script, the name of the script) as the positional parameter numbered 0 (see **Command Search and Execution** on page 47).

663

664

7. The shell optionally waits for the command to complete and collects the exit status (see Section 2.8.2 on page 44).

2.2 Quoting

Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to preserve the literal meaning of the special characters in the next paragraph; prevent reserved words from being recognised as such; and prevent parameter expansion and command substitution within here-document processing (see Section 2.7.4 on page 41).

The following characters must be quoted if they are to represent themselves:

| & ; < > () \$ ' \ " ' <space> <tab> <newline>

and the following may need to be quoted under certain circumstances. That is, these characters may be special depending on conditions described elsewhere in this specification:

* ? [# ~ = %

The various quoting mechanisms are the escape character, single-quotes and double-quotes. The here-document represents another form of quoting; see Section 2.7.4 on page 41.

2.2.1 Escape Character (Backslash)

A backslash that is not quoted preserves the literal value of the following character, with the exception of a newline character. If a newline character follows the backslash, the shell will interpret this as line continuation. The backslash and newline characters will be removed before splitting the input into tokens. Since the escaped newline character is removed entirely from the input and is not replaced by any white space, it cannot serve as a token separator.

2.2.2 Single-quotes

Enclosing characters in single-quotes (' ') preserves the literal value of each character within the single-quotes. A single-quote cannot occur within single-quotes.

A backslash cannot be used to escape a single-quote in a single-quoted string. An embedded quote can be created by writing, for example: 'a\'b', which yields a'b. (See Section 2.6.5 on page 38 for a better understanding of how portions of words are either split into fields or remain concatenated.) A single token can be made up of concatenated partial strings containing all three kinds of quoting or escaping, thus permitting any combination of characters.

2.2.3 Double-quotes

Enclosing characters in double-quotes (" ") preserves the literal value of all characters within the double-quotes, with the exception of the characters dollar-sign, backquote and backslash, as follows:

\$ The dollar-sign retains its special meaning introducing parameter expansion (see Section 2.6.2 on page 33), a form of command substitution (see Section 2.6.3 on page 36), and arithmetic expansion (see Section 2.6.4 on page 37).

The input characters within the quoted string that are also enclosed between "\$(" and the matching "(" will not be affected by the double-quotes, but rather define that command whose output replaces the \$(...) when the word is expanded. The tokenising rules in Section 2.3 on page 23 are applied recursively to find the matching ")".

Within the string of characters from an enclosed \${ to the matching "}", an even number of unescaped double-quotes or single-quotes, if any, must occur. A preceding backslash character must be used to escape a literal "{" or "}". The rule in Section 2.6.2 on page 33 will be used to determine the matching "}".

707 ` The backquote retains its special meaning introducing the other form of command
 708 substitution (see Section 2.6.3 on page 36). The portion of the quoted string from the initial
 709 backquote and the characters up to the next backquote that is not preceded by a backslash,
 710 having escape characters removed, defines that command whose output replaces ` . . . `
 711 when the word is expanded. Either of the following cases produces undefined results:

- 712 • a single- or double-quoted string that begins, but does not end, within the ` . . . `
 713 sequence
- 714 • a ` . . . ` sequence that begins, but does not end, within the same double-quoted
 715 string.

716 \ The backslash retains its special meaning as an escape character (see Section 2.2.1 on page
 717 20) only when followed by one of the characters:

718 \$ ` " \ <newline>

719 A double-quote must be preceded by a backslash to be included within double-quotes. The
 720 parameter @ has special meaning inside double-quotes and is described in Section 2.5.2 on page
 721 27.

722 In double-quoting, if a backslash is immediately followed by a character that would be
 723 interpreted as having a special meaning, the backslash is deleted and the subsequent character is
 724 taken literally. If a backslash does not precede a character that would have a special meaning, it
 725 is left in place unmodified and the character immediately following it is also left unmodified.
 726 Thus, for example:

727 "\\$" → \$

728 "\a" → \a

729 The requirement that double-quotes be matched inside \${...} within double-quotes and the rule
 730 for finding the matching ")" in Section 2.6.2 on page 33 eliminate several subtle inconsistencies in
 731 expansion for historical shells in rare cases; for example:

732 "\${foo-bar}"

733 yields **bar** when **foo** is not defined, and is an invalid substitution when **foo** is defined, in many
 734 historical shells. The differences in processing the "\${...}" form have led to inconsistencies
 735 between historical systems. A consequence of this rule is that single-quotes cannot be used to
 736 quote the ")" within "\${...}"; for example:

737 unset bar
 738 foo="\${bar-' }' "

739 is invalid because the "\${...}" substitution contains an unpaired unescaped single-quote. The
 740 backslash can be used to escape the ")" in this example to achieve the desired result:

741 unset bar
 742 foo="\${bar-\} } "

Some systems have allowed the end of the word to terminate the backquoted command substitution, such as in:

```
"`echo hello"
```

This usage is undefined; the matching backquote is required by this specification. The other undefined usage can be illustrated by the example:

```
sh -c '` echo "foo`'
```

The description of the recursive actions involving command substitution can be illustrated with an example. Upon recognising the introduction of command substitution, the shell must parse input (in a new context), gathering the source for the command substitution until an unbalanced ")" or ` is located. For example, in the following:

```
echo "$ (date; echo "
      one" ) "
```

the double-quote following the *echo* does not terminate the first double-quote; it is part of the command substitution script. Similarly, in:

```
echo "$ (echo * ) "
```

the asterisk is not quoted since it is inside command substitution; however:

```
echo "$ (echo " * " ) "
```

is quoted (and represents the asterisk character itself).

2.3 Token Recognition

The shell reads its input in terms of lines from a file, from a terminal in the case of an interactive shell or from a string in the case of *sh -c* or *system()*. The input lines can be of unlimited length. These lines are parsed using two major modes: ordinary token recognition and processing of here-documents.

When an **io_here** token has been recognised by the grammar (see Section 2.10 on page 56), one or more of the subsequent lines immediately following the next **NEWLINE** token form the body of one or more here-documents and are parsed according to the rules of Section 2.7.4 on page 41.

When it is not processing an **io_here**, the shell will break its input into tokens by applying the first applicable rule below to the next character in its input. The token will be from the current position in the input until a token is delimited according to one of the rules below; the characters forming the token are exactly those in the input, including any quoting characters. If it is indicated that a token is delimited, and no characters have been included in a token, processing will continue until an actual token is delimited.

1. If the end of input is recognised, the current token will be delimited. If there is no current token, the end-of-input indicator will be returned as the token.
2. If the previous character was used as part of an operator and the current character is not quoted and can be used with the current characters to form an operator, it will be used as part of that (operator) token.

Note that certain combinations of characters are invalid in portable scripts, as shown in the grammar, and that some systems have assigned these combinations (such as `|&`) as valid control operators. Portable scripts cannot rely on receiving errors in all cases where this specification indicates that a syntax is invalid.

3. If the previous character was used as part of an operator and the current character cannot be used with the current characters to form an operator, the operator containing the previous character will be delimited.
4. If the current character is backslash, single-quote or double-quote (`\`, `'` or `"`) and it is not quoted, it will affect quoting for subsequent characters up to the end of the quoted text. The rules for quoting are as described in Section 2.2 on page 20. During token recognition no substitutions will be actually performed, and the result token will contain exactly the characters that appear in the input (except for newline character joining), unmodified, including any embedded or enclosing quotes or substitution operators, between the quote mark and the end of the quoted text. The token will not be delimited by the end of the quoted field.
5. If the current character is an unquoted `"` or ```, the shell will identify the start of any candidates for parameter expansion (Section 2.6.2 on page 33), command substitution (Section 2.6.3 on page 36), or arithmetic expansion (Section 2.6.4 on page 37) from their introductory unquoted character sequences: `"$"` or `${`, `$(` or ```, and `$(`, respectively. The shell will read sufficient input to determine the end of the unit to be expanded (as explained in the cited sections). While processing the characters, if instances of expansions or quoting are found nested within the substitution, the shell will recursively process them in the manner specified for the construct that is found. The characters found from the beginning of the substitution to its end, allowing for any recursion necessary to recognise embedded constructs, will be included unmodified in the result token, including any embedded or enclosing substitution operators or quotes. The token will not be delimited by the end of the substitution.

6. If the current character is not quoted and can be used as the first character of a new operator, the current token (if any) will be delimited. The current character will be used as the beginning of the next (operator) token.
7. If the current character is an unquoted newline character, the current token will be delimited.
8. If the current character is an unquoted blank character, any token containing the previous character is delimited and the current character will be discarded.
9. If the previous character was part of a word, the current character will be appended to that word.
10. If the current character is a "#", it and all subsequent characters up to, but excluding, the next newline character will be discarded as a comment. The newline character that ends the line is not considered part of the comment. The "#" starts a comment only when it is at the beginning of a token. Since the search for the end-of-comment does not consider an escaped newline character specially, a comment cannot be continued to the next line.
11. The current character will be used as the start of a new word.

Once a token is delimited, it will be categorised as required by the grammar in Section 2.10 on page 56.

2.3.1 Alias Substitution

The processing of aliases is supported on all XSI-conformant systems.

After a token has been delimited, but before applying the grammatical rules in Section 2.10 on page 56, a resulting word that is identified to be the command name word of a simple command is examined to determine if it is an unquoted, valid alias name. However, reserved words in correct grammatical context are not candidates for alias substitution. A valid alias name (see the term *alias name* in the **XBD** specification, **Chapter 2, Glossary**) is one that has been defined by the *alias* utility and not subsequently undefined using *unalias*. Implementations also may provide predefined valid aliases that are in effect when the shell is invoked. To prevent infinite loops in recursive aliasing, if the shell is not currently processing an alias of the same name, the word will be replaced by the value of the alias; otherwise, it will not be replaced.

If the value of the alias replacing the word ends in a blank character, the shell will check the next command word for alias substitution; this process continues until a word is found that is not a valid alias or an alias value does not end in a blank character.

When used as specified by this specification, alias definitions will not be inherited by separate invocations of the shell or by the utility execution environments invoked by the shell; see Section 2.12 on page 63.

The definition of *alias name* precludes an alias name containing a slash character. Since the text applies to the command words of simple commands, reserved words (in their proper places) cannot be confused with aliases.

An example concerning trailing blank characters and reserved words follows. If the user types:

```
$ alias foo="/bin/ls "  
$ alias while="/ "
```


847 The effect of executing:

```
848       $ while true
849       > do
850       > echo "Hello, World"
851       > done
```

852 is a never-ending sequence of **Hello, World** strings to the screen. However, if the user types:

```
853       $ foo while
```

854 the result will be an *ls* listing of /. Since the alias substitution for **foo** ends in a space character,
855 the next word is checked for alias substitution. The next word, **while**, has also been aliased, so it
856 is substituted as well. Since it is not in the proper position as a command word, it is not
857 recognised as a reserved word.

858 If the user types:

```
859       $ foo; while
```

860 **while** retains its normal reserved-word properties.

2.4 Reserved Words

Reserved words are words that have special meaning to the shell. (See Section 2.9 on page 45.)
The following words will be recognised as reserved words:

!	elif	fi	in	while
case	else	for	then	{*
do	esac	if	until	}
done				

This recognition will occur only when none of the characters are quoted and when the word is used as:

- the first word of a command
- the first word following one of the reserved words other than **case**, **for**, or **in**
- the third word in a **case** or **for** command (only **in** is valid in this case).

See the grammar in Section 2.10 on page 56.

The following words may be recognised as reserved words on some systems (when none of the characters are quoted), causing unspecified results:

function	select	[[]]
-----------------	---------------	-----------	-----------

This list of unspecified reserved words is from the KornShell, so portable applications cannot use them in places a reserved word would be recognised without quoting some or all of their characters.

Words that are the concatenation of a name and a colon (:) are reserved; their use produces unspecified results. This reservation is to allow future implementations that support named labels for flow control.

Reserved words are recognised only when they are delimited (that is, meet the definition of **word** in the **XSH** specification), whereas operators are themselves delimiters. For instance, "(" and ")" are control operators, so that no space character is needed in (list). However, "{" and "}" are reserved words in { list;}, so that in this case the leading space character and semicolon are required.

* In some historical systems, the curly braces are treated as control operators. To assist in future standardisation activities, portable applications should avoid using unquoted braces to represent the characters themselves. It is possible that a future version of the ISO/IEC 9945-2: 1993 standard may require that { and } be treated individually as control operators, although the token {} will probably be a special-case exemption from this because of the often-used *find*{ } construct.

2.5 Parameters and Variables

A parameter can be denoted by a name, a number or one of the special characters listed in Section 2.5.2. A variable is a parameter denoted by a name.

A parameter is set if it has an assigned value (null is a valid value). Once a variable is set, it can only be unset by using the *unset* special built-in command.

2.5.1 Positional Parameters

A positional parameter is a parameter denoted by the decimal value represented by one or more digits, other than the single digit 0. The digits denoting the positional parameters are always interpreted as a decimal value, even if there is a leading zero. When a positional parameter with more than one digit is specified, the application must enclose the digits in braces (see Section 2.6.2 on page 33). Positional parameters are initially assigned when the shell is invoked (see *sh*), temporarily replaced when a shell function is invoked (see Section 2.9.5 on page 54), and can be reassigned with the *set* special built-in command.

2.5.2 Special Parameters

Listed below are the special parameters and the values to which they will expand. Only the values of the special parameters are listed; see Section 2.6 on page 31 for a detailed summary of all the stages involved in expanding words.

- * Expands to the positional parameters, starting from one. When the expansion occurs within a double-quoted string (see Section 2.2.3 on page 20), it expands to a single field with the value of each parameter separated by the first character of the *IFS* variable, or by a space character if *IFS* is unset. If *IFS* is set to a null string, this is not equivalent to unsetting it; its first character will not exist, so the parameter values are concatenated. For example:

```
$ IFS=' '
$ set foo bar bam
$ echo "$@"
foo bar bam
$ echo "$*"
foobarbam
$ unset IFS
$ echo "$*"
foo bar bam
```

- @ Expands to the positional parameters, starting from one. When the expansion occurs within double-quotes, and where field splitting (see Section 2.6.5 on page 38) is performed, each positional parameter expands as a separate field, with the provision that the expansion of the first parameter is still joined with the beginning part of the original word (assuming that the expanded parameter was embedded within a word), and the expansion of the last parameter is still joined with the last part of the original word. If there are no positional parameters, the expansion of "@" generates zero fields, even when "@" is double-quoted.

- # Expands to the decimal number of positional parameters. The command name (parameter 0) is not counted in the number given by "#" because it is a special parameter, not a positional parameter.

- ? Expands to the decimal exit status of the most recent pipeline (see Section 2.9.2 on page 49).
- (Hyphen.) Expands to the current option flags (the single-letter option names concatenated into a string) as specified on invocation, by the *set* special built-in command or implicitly by the shell.

The `$-` special parameter can be used to save and restore *set* options:

```
Save=$(echo $- | sed 's/[ics]//g')
...
set +aCefnuvx
if [ -n "$Save" ]; then
    set -$Save
fi
```

The three options are removed using *sed* in the example because they may appear in the value of `$-` (from the *sh* command line), but are not valid options to *set*.

\$ Expands to the decimal process ID of the invoked shell. In a subshell (see Section 2.12 on page 63), "\$" expands to the same value as that of the current shell.

Most historical implementations implement subshells by forking; thus, the special parameter "\$" does not necessarily represent the process ID of the shell process executing the commands since the subshell execution environment preserves the value of "\$".

! Expands to the decimal process ID of the most recent background command (see Section 2.9.3 on page 49) executed from the current shell. (For example, background commands executed from subshells do not affect the value of \$! in the current shell environment.) For a pipeline, the process ID is that of the last command in the pipeline.

0 (Zero.) Expands to the name of the shell or shell script. See *sh* for a detailed description of how this name is derived.

See the description of the *IFS* variable in Section 2.5.3 on page 29.

The descriptions of parameters "*" and "@" assume the reader is familiar with the field splitting discussion in Section 2.6.5 on page 38 and understands that portions of the word will remain concatenated unless there is some reason to split them into separate fields.

Some examples of the "*" and "@" properties, including the concatenation aspects:

```
set "abc" "def ghi" "jkl"

echo $*      => "abc" "def" "ghi" "jkl"
echo "$*"    => "abc def ghi jkl"
echo $@      => "abc" "def" "ghi" "jkl"
```

but:

```
echo "$@"      => "abc" "def ghi" "jkl"
echo "xx$@yy"  => "xxabc" "def ghi" "jkl"yy"
echo "$@$@"    => "abc" "def ghi" "jklabc" "def ghi" "jkl"
```

In the preceding examples, the double-quote characters that appear after the => do not appear in the output and are used only to illustrate word boundaries.

2.5.3 Shell Variables

Variables are initialised from the environment (as defined by the **XSH** specification) and can be given new values with variable assignment commands. If a variable is initialised from the environment, it is marked for export immediately; see the *export* special built-in. New variables can be defined and initialised with variable assignments, with the *read* or *getopts* utilities, with the *name* parameter in a **for** loop, with the `${name=word}` expansion or with other mechanisms provided as implementation extensions. The following variables affect the execution of the shell:

ENV This variable, when the shell is invoked, is subjected to parameter expansion (see Section 2.6.2 on page 33) by the shell and the resulting value is used as a pathname of a file containing shell commands to execute in the current environment. The file need not be executable. If the expanded value of *ENV* is not an absolute pathname, the results are unspecified. *ENV* will be ignored if the user's real and effective user IDs or real and effective group IDs are different.

This variable can be used to set aliases and other items local to the invocation of a shell. The file referred to by *ENV* differs from `$HOME/.profile` in that `.profile` is typically executed at session startup, whereas the *ENV* file is executed at the beginning of each shell invocation. The *ENV* value is interpreted in a manner similar to a dot script, in that the commands are executed in the current environment and the file needs to be readable, but not executable. However, unlike dot scripts, no *PATH* searching is performed. This is used as a guard against Trojan Horse security breaches.

HOME This variable is interpreted as the pathname of the user's home directory. The contents of *HOME* are used in tilde expansion (see Section 2.6.1 on page 32).

IFS *Input field separators*: a string treated as a list of characters that is used for field splitting and to split lines into fields with the *read* command. If *IFS* is not set, the shell will behave as if the value of *IFS* were the space, tab and newline characters. (See Section 2.6.5 on page 38.)

LANG Provide a default value for the internationalisation variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL This variable provides a default value for the *LC_** variables, as described in the **XBD** specification, **Chapter 6, Environment Variables**.

LC_COLLATE This variable determines the behaviour of range expressions, equivalence classes and multi-character collating elements within pattern matching.

LC_CTYPE This variable determines the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters), which characters are defined as letters (character class **alpha**) and blank characters (character class **blank**), and the behaviour of character classes within pattern matching. Changing the value of *LC_CTYPE* after the shell has started does not affect the lexical processing of shell commands in the current shell execution environment or its subshells. Invoking a shell script or performing *exec sh* subjects the new shell to the changes in *LC_CTYPE*.

1019	<i>LC_MESSAGES</i>	This variable determines the language in which messages should be written.
1020	<i>LINENO</i>	This variable is set by the shell to a decimal number representing the current sequential line number (numbered starting with 1) within a script or function before it executes each command. If the user unsets or resets <i>LINENO</i> , the variable may lose its special meaning for the life of the shell. If the shell is not currently executing a script or function, the value of <i>LINENO</i> is unspecified.
1021		
1022		
1023		
1024		
1025 EX	<i>NLSPATH</i>	Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .
1026		
1027	<i>PATH</i>	This variable represents a string formatted as described in the XBD specification, Chapter 6, Environment Variables , used to effect command interpretation. See Command Search and Execution on page 47.
1028		
1029		
1030	<i>PPID</i>	This variable is set by the shell to the decimal process ID of the process that invoked this shell. In a subshell (see Section 2.12 on page 63), <i>PPID</i> will be set to the same value as that of the parent of the current shell. For example, <i>echo \$PPID</i> and (<i>echo \$PPID</i>) would produce the same value.
1031		
1032		
1033		
1034		
1035		Without this variable, there is no way for a utility to signal its parent or to find its parent process. This is also useful to know if the shell has been orphaned.
1036	<i>PS1</i>	Each time an interactive shell is ready to read a command, the value of this variable is subjected to parameter expansion and written to standard error. The default value is "\$ ". For users who have specific additional implementation-dependent privileges, the default may be another, implementation-dependent, value. (Historically, the superuser has had a prompt of "# "). The shell replaces each instance of the character "!" in <i>PS1</i> with the history file number of the next command to be typed. Escaping the "!" with another "!" (that is, !!) places the literal character "!" in the prompt.
1037		
1038		
1039		
1040		
1041		
1042		
1043		
1044	<i>PS2</i>	Each time the user enters a newline character prior to completing a command line in an interactive shell, the value of this variable is subjected to parameter expansion and written to standard error. The default value is "> ".
1045		
1046		
1047	<i>PS4</i>	When an execution trace (<i>set -x</i>) is being performed in an interactive shell, before each line in the execution trace, the value of this variable is subjected to parameter expansion and written to standard error. The default value is "+ ".
1048		
1049		
1050		For example, the following script:
1051		<code>PS4=' [\${LINENO}]+ '</code>
1052		<code>set -x</code>
1053		<code>echo Hello</code>
1054		writes the following to standard error:
1055		<code>[3]+ echo Hello</code>
1056		Tilde expansion for components of the <i>PATH</i> in an assignment such as:
1057		<code>PATH=~hlj/bin:~dwc/bin:\$PATH</code>
1058		is a feature of some historical shells and is allowed by the wording of Section 2.6.1 on page 32.
1059		Note that the tildes are expanded during the assignment to <i>PATH</i> , not when <i>PATH</i> is accessed
1060		during command search.

2.6 Word Expansions

This section describes the various expansions that are performed on words. Not all expansions are performed on every word, as explained in the following sections.

Tilde expansions, parameter expansions, command substitutions, arithmetic expansions and quote removals that occur within a single word expand to a single field. It is only field splitting or pathname expansion that can create multiple fields from a single word. The single exception to this rule is the expansion of the special parameter "@" within double-quotes, as described in Section 2.5.2 on page 27.

The order of word expansion is as follows:

1. Tilde expansion (see Section 2.6.1 on page 32), parameter expansion (see Section 2.6.2 on page 33), command substitution (see Section 2.6.3 on page 36), and arithmetic expansion (see Section 2.6.4 on page 37) are performed, beginning to end. See item 5 in Section 2.3 on page 23.
2. Field splitting (see Section 2.6.5 on page 38) is performed on the portions of the fields generated by step 1, unless *IFS* is null.
3. Pathname expansion (see Section 2.6.6 on page 39) is performed, unless *set -f* is in effect.
4. Quote removal (see Section 2.6.7 on page 39) always is performed last.

The expansions described in this section will occur in the same shell environment as that in which the command is executed.

If the complete expansion appropriate for a word results in an empty field, that empty field will be deleted from the list of fields that form the completely expanded command, unless the original word contained single-quote or double-quote characters.

The "\$" character is used to introduce parameter expansion, command substitution or arithmetic evaluation. If an unquoted "\$" is followed by a character that is either not numeric, the name of one of the special parameters (see Section 2.5.2 on page 27), a valid first character of a variable name, a left curly brace ({}), or a left parenthesis, the result is unspecified.

IFS is used for performing field splitting on the results of parameter and command substitution; it is not used for splitting all fields. Previous versions of the shell used it for splitting all fields during field splitting, but this has severe problems because the shell can no longer parse its own script. There are also important security implications caused by this behaviour. All useful applications of *IFS* use it for parsing input of the *read* utility and for splitting the results of parameter and command substitution.

The rule concerning expansion to a single field requires that if **foo=abc** and **bar=def**, that:

```
"$foo" "$bar"
```

expands to the single field:

```
abcdef
```

1097 The rule concerning empty fields can be illustrated by:

```
1098 $ unset foo
1099 $ set $foo bar ' ' xyz "$foo" abc
1100 $ for i
1101 > do
1102 >     echo "-$i-"
1103 > done
1104 -bar-
1105 - -
1106 -xyz-
1107 - -
1108 -abc-
```

1109 Step 2 indicates that parameter expansion, command substitution and arithmetic expansion are
1110 all processed simultaneously as they are scanned. For example, the following is valid arithmetic:

```
1111 x=1
1112 echo $(( $(echo 3)+$x ))
```

1113 2.6.1 Tilde Expansion

1114 A *tilde-prefix* consists of an unquoted tilde character at the beginning of a word, followed by all
1115 of the characters preceding the first unquoted slash in the word, or all the characters in the word
1116 if there is no slash. In an assignment (see **variable assignment** in the **XBD** specification,
1117 **Chapter 2, Glossary**) multiple tilde-prefixes can be used: at the beginning of the word (that is,
1118 following the equal sign of the assignment), following any unquoted colon or both. A tilde-
1119 prefix in an assignment is terminated by the first unquoted colon or slash. If none of the
1120 characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the tilde are
1121 treated as a possible login name from the user database. A portable login name cannot contain
1122 characters outside the set given in the description of the *LOGNAME* environment variable in the
1123 **XSH** specification. If the login name is null (that is, the tilde-prefix contains only the tilde), the
1124 tilde-prefix will be replaced by the value of the variable *HOME*. If *HOME* is unset, the results are
1125 unspecified. Otherwise, the tilde-prefix will be replaced by a pathname of the home directory
1126 associated with the login name obtained using the **XSH** specification *getpwnam()* function. If the
1127 system does not recognise the login name, the results are undefined.

1128 Tilde expansion generally occurs only at the beginning of words, but an exception based on
1129 historical practice has been included:

```
1130 PATH=/posix/bin:~dgk/bin
```

1131 is eligible for tilde expansion because tilde follows a colon and none of the relevant characters is
1132 quoted. Consideration was given to prohibiting this behaviour because any of the following are
1133 reasonable substitutes:

```
1134 OB PATH=$(printf %s: ~rms/bin ~bfox/bin ...)
```

```
1135 PATH=$(printf %s ~karels/bin : ~bostic/bin)
```

```
1136 for Dir in ~maat/bin ~srb/bin ...
```

```
1137 do
```

```
1138     PATH=${PATH:+$PATH:}$Dir
```

```
1139 done
```

1140 OB In the first command, any number of directory names are concatenated and separated with
1141 colons, but it may be undesirable to end the variable with a colon because this is an obsolescent
1142 means to include dot at the end of the *PATH*. In the second, explicit colons are used for each

1143 directory. In all cases, the shell performs tilde expansion on each directory because all are
 1144 separate words to the shell.

1145 Note that expressions in operands such as:

1146 `make -k mumble LIBDIR=~chet/lib`

1147 do not qualify as shell variable assignments and tilde expansion is not performed (unless the
 1148 command does so itself, which *make* does not).

1149 The special sequence `$~` has been designated for future implementations to evaluate as a means
 1150 of forcing tilde expansion in any word.

1151 Because of the requirement that the word is not quoted, the following are not equivalent; only
 1152 the last will cause tilde expansion:

1153 `\~hlj/` `~h\lj/` `~"hlj/"` `~hlj\` `~hlj/`

1154 The results of giving tilde with an unknown login name are undefined because the KornShell `~+`
 1155 and `~-` constructs make use of this condition, but, in general it is an error to give an incorrect
 1156 login name with tilde. The results of having *HOME* unset are unspecified because some
 1157 historical shells treat this as an error.

1158 2.6.2 Parameter Expansion

1159 The format for parameter expansion is as follows:

1160 `${expression}`

1161 where *expression* consists of all characters until the matching `}`. Any `"}` escaped by a backslash
 1162 or within a quoted string, and characters in embedded arithmetic expansions, command
 1163 substitutions and variable expansions, are not examined in determining the matching `"}`.

1164 The simplest form for parameter expansion is:

1165 `${parameter}`

1166 The value, if any, of *parameter* will be substituted.

1167 The parameter name or symbol can be enclosed in braces, which are optional except for
 1168 positional parameters with more than one digit or when *parameter* is followed by a character that
 1169 could be interpreted as part of the name. The matching closing brace will be determined by
 1170 counting brace levels, skipping over enclosed quoted strings and command substitutions.

1171 If the parameter name or symbol is not enclosed in braces, the expansion will use the longest
 1172 valid name (see **name** in the **XBD** specification, **Chapter 2, Glossary**), whether or not the symbol
 1173 represented by that name exists. When the shell is scanning its input to determine the
 1174 boundaries of a name, it is not bound by its knowledge of what names are already defined. For
 1175 example, if **F** is a defined shell variable, the command:

1176 `echo $Fred`

1177 does not echo the value of **\$F** followed by **red**; it selects the longest possible valid name, **Fred**,
 1178 which in this case might be unset.

1179 If a parameter expansion occurs inside double-quotes:

- 1180 • Pathname expansion will not be performed on the results of the expansion.
- 1181 • Field splitting will not be performed on the results of the expansion, with the exception of
 1182 `"@"`; see Section 2.5.2 on page 27.

In addition, a parameter expansion can be modified by using one of the following formats. In each case that a value of *word* is needed (based on the state of *parameter*, as described below), *word* will be subjected to tilde expansion, parameter expansion, command substitution and arithmetic expansion. If *word* is not needed, it will not be expanded. The `"}` character that delimits the following parameter expansion modifications is determined as described previously in this section and in Section 2.2.3 on page 20. (For example, `${foo-bar}xyz` would result in the expansion of `foo` followed by the string `xyz` if `foo` is set, else the string `barxyz`).

`${parameter:-word}` **Use Default Values.** If *parameter* is unset or null, the expansion of *word* will be substituted; otherwise, the value of *parameter* will be substituted.

`${parameter:=word}` **Assign Default Values.** If *parameter* is unset or null, the expansion of *word* will be assigned to *parameter*. In all cases, the final value of *parameter* will be substituted. Only variables, not positional parameters or special parameters, can be assigned in this way.

`${parameter:?[word]}` **Indicate Error if Null or Unset.** If *parameter* is unset or null, the expansion of *word* (or a message indicating it is unset if *word* is omitted) will be written to standard error and the shell will exit with a non-zero exit status. Otherwise, the value of *parameter* will be substituted. An interactive shell need not exit.

`${parameter:+word}` **Use Alternative Value.** If *parameter* is unset or null, null will be substituted; otherwise, the expansion of *word* will be substituted.

In the parameter expansions shown previously, use of the colon in the format results in a test for a parameter that is unset or null; omission of the colon results in a test for a parameter that is only unset. The following table summarises the effect of the colon:

	parameter set and not null	parameter set but null	parameter unset
<code>\${parameter:-word}</code>	substitute <i>parameter</i>	substitute <i>word</i>	substitute <i>word</i>
<code>\${parameter-word}</code>	substitute <i>parameter</i>	substitute null	substitute <i>word</i>
<code>\${parameter:=word}</code>	substitute <i>parameter</i>	assign <i>word</i>	assign <i>word</i>
<code>\${parameter=word}</code>	substitute <i>parameter</i>	substitute <i>parameter</i>	assign null
<code>\${parameter:?word}</code>	substitute <i>parameter</i>	error, exit	error, exit
<code>\${parameter?word}</code>	substitute <i>parameter</i>	substitute null	error, exit
<code>\${parameter:+word}</code>	substitute <i>word</i>	substitute null	substitute null
<code>\${parameter+word}</code>	substitute <i>word</i>	substitute <i>word</i>	substitute null

In all cases shown with “substitute”, the expression is replaced with the value shown. In all cases shown with “assign”, *parameter* is assigned that value, which also replaces the expression.

`${#parameter}` **String Length.** The length in characters of the value of *parameter*. If *parameter* is `"*"` or `"@"`, the result of the expansion is unspecified.

The following four varieties of parameter expansion provide for substring processing. In each case, pattern matching notation (see Section 2.13 on page 64), rather than regular expression notation, will be used to evaluate the patterns. If *parameter* is `"*"` or `"@"`, the result of the expansion is unspecified. Enclosing the full parameter expansion string in double-quotes will not cause the following four varieties of pattern characters to be quoted, whereas quoting characters within the braces will have this effect.

`${parameter%word}` **Remove Smallest Suffix Pattern.** The *word* will be expanded to produce a pattern. The parameter expansion then will result in *parameter*, with the smallest portion of the suffix matched by the *pattern* deleted.

1230 `${parameter%%word}` **Remove Largest Suffix Pattern.** The *word* will be expanded to produce a
 1231 pattern. The parameter expansion then will result in *parameter*, with the
 1232 largest portion of the suffix matched by the *pattern* deleted.

1233 `${parameter#word}` **Remove Smallest Prefix Pattern.** The *word* will be expanded to produce a
 1234 pattern. The parameter expansion then will result in *parameter*, with the
 1235 smallest portion of the prefix matched by the *pattern* deleted.

1236 `${parameter##word}` **Remove Largest Prefix Pattern.** The *word* will be expanded to produce a
 1237 pattern. The parameter expansion then will result in *parameter*, with the
 1238 largest portion of the prefix matched by the *pattern* deleted.

1239 Examples

1240 `${parameter:-word}`
 1241 In this example, *ls* is executed only if *x* is null or unset. (The `$(ls)` command substitution
 1242 notation is explained in Section 2.6.3 on page 36.)

1243 `${x:-$(ls)}`

1244 `${parameter:=word}`

1245 `unset X`
 1246 `echo ${X:=abc}`
 1247 **abc**

1248 `${parameter:?word}`

1249 `unset posix`
 1250 `echo ${posix:?}`
 1251 **sh: posix: parameter null or not set**

1252 `${parameter:+word}`

1253 `set a b c`
 1254 `echo ${3:+posix}`
 1255 **posix**

1256 `${#parameter}`

1257 `HOME=/usr/posix`
 1258 `echo ${#HOME}`
 1259 **10**

1260 `${parameter%word}`

1261 `x=file.c`
 1262 `echo ${x%.c}.o`
 1263 **file.o**

1264 `${parameter%%word}`

1265 `x=posix/src/std`
 1266 `echo ${x%%/*}`
 1267 **posix**

1268 `${parameter#word}`

```
1269         x=$HOME/src/cmd
1270         echo ${x#$HOME}
1271         /src/cmd
```

1272 `${parameter##word}`

```
1273         x=/one/two/three
1274         echo ${x##*/}
1275         three
```

1276 The double-quoting of patterns is different depending on where the double-quotes are placed:

1277 `"${x#*}"` The asterisk is a pattern character.

1278 `${x#"*}"` The literal asterisk is quoted and not special.

1279 2.6.3 Command Substitution

1280 Command substitution allows the output of a command to be substituted in place of the
1281 command name itself. Command substitution occurs when the command is enclosed as follows:

1282 `$(command)`

1283 or (backquoted version):

1284 ``command``

1285 The shell will expand the command substitution by executing *command* in a subshell
1286 environment (see Section 2.12 on page 63) and replacing the command substitution (the text of
1287 *command* plus the enclosing `$()` or backquotes) with the standard output of the command,
1288 removing sequences of one or more newline characters at the end of the substitution. Embedded
1289 newline characters before the end of the output will not be removed; however, they may be
1290 treated as field delimiters and eliminated during field splitting, depending on the value of *IFS*
1291 and quoting that is in effect.

1292 Within the backquoted style of command substitution, backslash shall retain its literal meaning,
1293 except when followed by:

1294 `$ ` \`

1295 (dollar-sign, backquote, backslash). The search for the matching backquote is satisfied by the
1296 first backquote found without a preceding backslash; during this search, if a non-escaped
1297 backquote is encountered within a shell comment, a here-document, an embedded command
1298 substitution of the `$(command)` form, or a quoted string, undefined results occur. A single- or
1299 double-quoted string that begins, but does not end, within the `` . . . `` sequence produces
1300 undefined results.

1301 With the `$(command)` form, all characters following the open parenthesis to the matching closing
1302 parenthesis constitute the *command*. Any valid shell script can be used for *command*, except:

- 1303 • a script consisting solely of redirections produces unspecified results
- 1304 • see the restriction on single subshells described below.

1305 The results of command substitution will not be field splitting and pathname expansion
1306 processed for further tilde expansion, parameter expansion, command substitution or arithmetic
1307 expansion. If a command substitution occurs inside double-quotes, it will not be performed on
1308 the results of the substitution.

Command substitution can be nested. To specify nesting within the backquoted version, the application must precede the inner backquotes with backslashes; for example:

```
\`command\`
```

The \$() form of command substitution solves a problem of inconsistent behaviour when using backquotes. For example:

Command	Output
echo `\\$x`	\\$x
echo `echo `\\$x``	\$x
echo \$(echo `\\$x`)	\\$x

Additionally, the backquoted syntax has historical restrictions on the contents of the embedded command. While the new \$() form can process any kind of valid embedded script, the backquoted form cannot handle some valid scripts that include backquotes. For example, these otherwise valid embedded scripts do not work in the left column, but do work on the right:

echo `	echo \$(
cat <<\eof	cat <<\eof
a here-doc with `	a here-doc with)
eof	eof
`)
echo `	echo \$(
echo abc # a comment with `	echo abc # a comment with)
`)
echo `	echo \$(
echo ` ` `	echo ` ` `
`)

Because of these inconsistent behaviours, the backquoted variety of command substitution is not recommended for new applications that nest command substitutions or attempt to embed complex scripts.

If the command substitution consists of a single subshell, such as:

```
$( (command) )
```

a portable application must separate the \$(and "(" into two tokens (that is, separate them with white space). This is required to avoid any ambiguities with arithmetic expansion.

2.6.4 Arithmetic Expansion

Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion is as follows:

```
$( (expression) )
```

The expression is treated as if it were in double-quotes, except that a double-quote inside the expression is not treated specially. The shell will expand all tokens in the expression for parameter expansion, command substitution and quote removal.

Next, the shell will treat this as an arithmetic expression and substitute the value of the expression. The arithmetic expression will be processed according to the rules of the ISO C standard, with the following exceptions:

- Only integer arithmetic is required.
- The `sizeof()` operator and the prefix and postfix `++` and `--` operators are not required.
- Selection, iteration and jump statements are not supported.

As an extension, the shell may recognise arithmetic expressions beyond those listed. If the expression is invalid, the expansion will fail and the shell will write a message to standard error indicating the failure.

A simple example using arithmetic expansion:

```
# repeat a command 100 times
x=100
while [ $x -gt 0 ]
do
    command
    x=$(( $x-1 ))
done
```

2.6.5 Field Splitting

After parameter expansion (Section 2.6.2 on page 33), command substitution (Section 2.6.3 on page 36), and arithmetic expansion (Section 2.6.4 on page 37) the shell will scan the results of expansions and substitutions that did not occur in double-quotes for field splitting and multiple fields can result.

The shell will treat each character of the *IFS* as a delimiter and use the delimiters to split the results of parameter expansion and command substitution into fields.

1. If the value of *IFS* is a space, tab and newline character, or if it is unset, any sequence of space, tab or newline characters at the beginning or end of the input will be ignored and any sequence of those characters within the input will delimit a field. For example, the input:

```
<newline><space><tab>foo<tab><tab>bar<space>
```

yields two fields, **foo** and **bar**.

2. If the value of *IFS* is null, no field splitting will be performed.
3. Otherwise, the following rules will be applied in sequence. The term “*IFS* white space” is used to mean any sequence (zero or more instances) of white-space characters that are in the *IFS* value (for example, if *IFS* contains space/comma/tab, any sequence of space and tab characters is considered *IFS* white space).
 - a. *IFS* white space is ignored at the beginning and end of the input.
 - b. Each occurrence in the input of an *IFS* character that is not *IFS* white space, along with any adjacent *IFS* white space, will delimit a field, as described previously.
 - c. Non-zero-length *IFS* white space will delimit a field.

The last rule can be summarised as a pseudo-ERE:

```
( s*ns* | s+ )
```

where *s* is an *IFS* white-space character and *n* is a character in the *IFS* that is not white space. Any string matching that ERE delimits a field, except that the *s+* form does not delimit fields at the beginning or the end of a line. For example, if *IFS* is space/comma/tab, the string:

1392 <space><space>red<space><space>,<space>white<space>blue

1393 yields the three colours as the delimited fields.

1394 **2.6.6 Pathname Expansion**

1395 After field splitting, if *set -f* is not in effect, each field in the resulting command line will be
1396 expanded using the algorithm described in Section 2.13 on page 64, qualified by the rules in
1397 Section 2.13.3 on page 66.

1398 **2.6.7 Quote Removal**

1399 The quote characters:

1400 \ ' "

1401 (backslash, single-quote, double-quote) that were present in the original word will be removed
1402 unless they have themselves been quoted.

2.7 Redirection

Redirection is used to open and close files for the current shell execution environment (see Section 2.12 on page 63) or for any command. *Redirection operators* can be used with numbers representing file descriptors (see the definition in the ISO POSIX-1 standard) as described below.

The overall format used for redirection is:

```
[n]redir-op word
```

The number *n* is an optional decimal number designating the file descriptor number; it must be delimited from any preceding text and immediately precede the redirection operator *redir-op*. If *n* is quoted, the number will not be recognised as part of the redirection expression. For example:

```
echo \2>a
```

writes the character 2 into file **a**. If any part of *redir-op* is quoted, no redirection expression will be recognised. For example:

```
echo 2\>a
```

writes the characters 2>a to standard output. The optional number, redirection operator and *word* will not appear in the arguments provided to the command to be executed (if any).

Open files are represented by decimal numbers starting with zero. The largest possible value is implementation-dependent; however, all implementations support at least 0 to 9, inclusive, for use by the application. These numbers are called *file descriptors*. The values 0, 1 and 2 have special meaning and conventional uses and are implied by certain redirection operations; they are referred to as *standard input*, *standard output* and *standard error*, respectively. Programs usually take their input from standard input, and write output on standard output. Error messages are usually written on standard error. The redirection operators can be preceded by one or more digits (with no intervening blank characters allowed) to designate the file descriptor number.

If the redirection operator is << or <<-, the word that follows the redirection operator will be subjected to quote removal; it is unspecified whether any of the other expansions occur. For the other redirection operators, the word that follows the redirection operator will be subjected to tilde expansion, parameter expansion, command substitution, arithmetic expansion and quote removal. Pathname expansion will not be performed on the word by a non-interactive shell; an interactive shell may perform it, but will do so only when the expansion would result in one word.

If more than one redirection operator is specified with a command, the order of evaluation is from beginning to end.

A failure to open or create a file will cause the redirection to fail.

2.7.1 Redirecting Input

Input redirection will cause the file whose name results from the expansion of *word* to be opened for reading on the designated file descriptor, or standard input if the file descriptor is not specified.

The general format for redirecting input is:

```
[n]<word
```

where the optional *n* represents the file descriptor number. If the number is omitted, the redirection will refer to standard input (file descriptor 0).

1446 2.7.2 Redirecting Output

1447 The two general formats for redirecting output are:

```
1448     [n]>word
1449     [n]> | word
```

1450 where the optional *n* represents the file descriptor number. If the number is omitted, the
1451 redirection will refer to standard output (file descriptor 1).

1452 Output redirection using the ">" format will fail if the *noclobber* option is set (see the description
1453 of *set -C*) and the file named by the expansion of *word* exists and is a regular file. Otherwise,
1454 redirection using the ">" or ">|" formats will cause the file whose name results from the expansion
1455 of *word* to be created and opened for output on the designated file descriptor, or standard output
1456 if none is specified. If the file does not exist, it will be created; otherwise, it will be truncated to
1457 be an empty file after being opened.

1458 2.7.3 Appending Redirected Output

1459 Appended output redirection will cause the file whose name results from the expansion of *word*
1460 to be opened for output on the designated file descriptor. The file is opened as if the **XSH**
1461 specification *open()* function was called with the *O_APPEND* flag. If the file does not exist, it
1462 will be created.

1463 The general format for appending redirected output is as follows:

```
1464     [n]>>word
```

1465 where the optional *n* represents the file descriptor number.

1466 2.7.4 Here-document

1467 The redirection operators << and <<- both allow redirection of lines contained in a shell input
1468 file, known as a *here-document*, to the standard input of a command.

1469 The here-document is treated as a single word that begins after the next newline character and
1470 continues until there is a line containing only the delimiter, with no trailing blank characters.
1471 Then the next here-document starts, if there is one. The format is as follows:

```
1472     [n]<<word
1473         here-document
1474     delimiter
```

1475 If any character in *word* is quoted, the delimiter is formed by performing quote removal on *word*,
1476 and the here-document lines will not be expanded. Otherwise, the delimiter is the *word* itself.

1477 If no characters in *word* are quoted, all lines of the here-document will be expanded for
1478 parameter expansion, command substitution and arithmetic expansion. In this case, the
1479 backslash in the input will behave as the backslash inside double-quotes (see Section 2.2.3 on
1480 page 20). However, the double-quote character (") will not be treated specially within a here-
1481 document, except when the double-quote appears within \$(), ` ` or \${ }.

1482 If the redirection symbol is <<-, all leading tab characters will be stripped from input lines and
 1483 the line containing the trailing delimiter. If more than one << or <<- operator is specified on a
 1484 line, the here-document associated with the first operator will be supplied first by the
 1485 application and will be read first by the shell. For example:

```
1486     cat <<eof1; cat <<eof2
1487     Hi ,
1488     eof1
1489     Helene.
1490     eof2
```

1491 The case of a missing delimiter at the end of a here-document is not specified. This is considered
 1492 an error in the script (one that sometimes can be difficult to diagnose), although some systems
 1493 have treated end-of-file as an implicit delimiter.

1494 2.7.5 Duplicating an Input File Descriptor

1495 The redirection operator:

```
1496     [n]<&word
```

1497 is used to duplicate one input file descriptor from another, or to close one. If *word* evaluates to
 1498 one or more digits, the file descriptor denoted by *n*, or standard input if *n* is not specified, will be
 1499 made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent a
 1500 file descriptor already open for input, a redirection error will result (see Section 2.8.1 on page 44).
 1501 If *word* evaluates to "-", file descriptor *n*, or standard input if *n* is not specified, will be closed. If
 1502 *word* evaluates to something else, the behaviour is unspecified.

1503 2.7.6 Duplicating an Output File Descriptor

1504 The redirection operator:

```
1505     [n]>&word
```

1506 is used to duplicate one output file descriptor from another, or to close one. If *word* evaluates to
 1507 one or more digits, the file descriptor denoted by *n*, or standard output if *n* is not specified, will
 1508 be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent
 1509 a file descriptor already open for output, a redirection error will result (see Section 2.8.1 on page
 1510 44). If *word* evaluates to "-", file descriptor *n*, or standard output if *n* is not specified, will be
 1511 closed. If *word* evaluates to something else, the behaviour is unspecified.

1512 The construct 2>&1 is often used to redirect standard error to the same file as standard output.
 1513 Since the redirections take place beginning to end, the order of redirections is significant. For
 1514 example:

```
1515     ls > foo 2>&1
```

1516 directs both standard output and standard error to file **foo**. However:

```
1517     ls 2>&1 > foo
```

1518 only directs standard output to file **foo** because standard error was duplicated as standard
 1519 output before standard output was directed to file **foo**.

1520 **2.7.7 Open File Descriptors for Reading and Writing**

1521 The redirection operator:

1522 `[n]<>word`

1523 will cause the file whose name is the expansion of *word* to be opened for both reading and
1524 writing on the file descriptor denoted by *n*, or standard input if *n* is not specified. If the file does
1525 not exist, it will be created.

1526 The <> operator could be useful in writing an application that worked with several terminals,
1527 and occasionally wanted to start up a shell. That shell would in turn be unable to run
1528 applications that run from an ordinary controlling terminal unless it could make use of <>
1529 redirection. The specific example is a historical version of the pager *more*, which reads from
1530 standard error to get its commands, so standard input and standard output are both available
1531 for their usual usage. There is no way of saying the following in the shell without <>:

1532 `cat food | more - >/dev/tty03 2<>/dev/tty03`1533 Another example of <> is one that opens **/dev/tty** on file descriptor 3 for reading and writing:1534 `exec 3<> /dev/tty`

1535 An example of creating a lock file for a critical code region:

```
1536 set -C
1537 until 2> /dev/null > lockfile
1538 do    sleep 30
1539 done
1540 set +C
1541 perform critical function
1542 rm lockfile
```

1543 Since **/dev/null** is not a regular file, no error is generated by redirecting to it in *noclobber* mode.

2.8 Exit Status and Errors

2.8.1 Consequences of Shell Errors

For a non-interactive shell, an error condition encountered by a special built-in (see Section 2.14 on page 67) or other type of utility will cause the shell to write a diagnostic message to standard error and exit as shown in the following table:

Error	Special Built-in	Other Utilities
Shell language syntax error	will exit	will exit
Utility syntax error (option or operand error)	will exit	will not exit
Redirection error	will exit	will not exit
Variable assignment error	will exit	will not exit
Expansion error	will exit	will exit
Command not found	n/a	may exit
Dot script not found	will exit	n/a

An expansion error is one that occurs when the shell expansions defined in Section 2.6 on page 31 are carried out (for example, `${x!y}`, because `!` is not a valid operator); an implementation may treat these as syntax errors if it is able to detect them during tokenisation, rather than during expansion.

If any of the errors shown as “will (may) exit” occur in a subshell, the subshell will (may) exit with a non-zero status, but the script containing the subshell will not exit because of the error.

In all of the cases shown in the table, an interactive shell will write a diagnostic message to standard error without exiting.

2.8.2 Exit Status for Commands

Each command has an exit status that can influence the behaviour of other shell commands. The exit status of commands that are not utilities is documented in this section. The exit status of the standard utilities is documented in their respective sections.

If a command is not found, the exit status will be 127. If the command name is found, but it is not an executable utility, the exit status will be 126. Applications that invoke utilities without using the shell should use these exit status values to report similar errors.

If a command fails during word expansion or redirection, its exit status will be greater than zero.

Internally, for purposes of deciding if a command exits with a non-zero exit status, the shell will recognise the entire status value retrieved for the command by the equivalent of the `XSH` specification `wait()` function `WEXITSTATUS` macro. When reporting the exit status with the special parameter `?`, the shell will report the full eight bits of exit status available. The exit status of a command that terminated because it received a signal will be reported as greater than 128.

1579 2.9 Shell Commands

1580 This section describes the basic structure of shell commands. The following command
 1581 descriptions each describe a format of the command that is only used to aid the reader in
 1582 recognising the command type, and does not formally represent the syntax. Each description
 1583 discusses the semantics of the command; for a formal definition of the command language,
 1584 consult Section 2.10 on page 56.

1585 A *command* is one of the following:

- 1586 • *simple command* (see Section 2.9.1)
- 1587 • *pipeline* (see Section 2.9.2 on page 49)
- 1588 • *list* or *compound-list* (see Section 2.9.3 on page 49)
- 1589 • *compound command* (see Section 2.9.4 on page 52)
- 1590 • *function definition* (see Section 2.9.5 on page 54).

1591 Unless otherwise stated, the exit status of a command is that of the last simple command
 1592 executed by the command. There is no limit on the size of any shell command other than that
 1593 imposed by the underlying system (memory constraints, {ARG_MAX}, and so on).

1594 2.9.1 Simple Commands

1595 A *simple command* is a sequence of optional variable assignments and redirections, in any
 1596 sequence, optionally followed by words and redirections, terminated by a control operator.

1597 When a given simple command is required to be executed (that is, when any conditional
 1598 construct such as an AND-OR list or a **case** statement has not bypassed the simple command),
 1599 the following expansions, assignments and redirections will all be performed from the beginning
 1600 of the command text to the end.

- 1601 1. The words that are recognised as variable assignments or redirections according to Section
 1602 2.10.2 on page 56 are saved for processing in steps 3 and 4.
- 1603 2. The words that are not variable assignments or redirections will be expanded. If any fields
 1604 remain following their expansion, the first field will be considered the command name and
 1605 remaining fields will be the arguments for the command.
- 1606 3. Redirections will be performed as described in Section 2.7 on page 40.
- 1607 4. Each variable assignment will be expanded for tilde expansion, parameter expansion,
 1608 command substitution, arithmetic expansion and quote removal prior to assigning the
 1609 value.

1610 In the preceding list, the order of steps 3 and 4 may be reversed for the processing of special
 1611 built-in utilities. See Section 2.14 on page 67.

1612 If no command name results, variable assignments will affect the current execution
 1613 environment. Otherwise, the variable assignments will be exported for the execution
 1614 environment of the command and will not affect the current execution environment (except for
 1615 special built-ins). If any of the variable assignments attempt to assign a value to a read-only
 1616 variable, a variable assignment error will occur. See Section 2.8.1 on page 44 for the
 1617 consequences of these errors.

1618 If there is no command name, any redirections will be performed in a subshell environment; it is
 1619 unspecified whether this subshell environment is the same one as that used for a command
 1620 substitution within the command. (To affect the current execution environment, see the *exec*
 1621 special built-in.) If any of the redirections performed in the current shell execution environment

fail, the command will immediately fail with an exit status greater than zero, and the shell will write an error message indicating the failure. See Section 2.8.1 on page 44 for the consequences of these failures on interactive and non-interactive shells.

If there is a command name, execution will continue as described in **Command Search and Execution** on page 47. If there is no command name, but the command contained a command substitution, the command will complete with the exit status of the last command substitution performed. Otherwise, the command will complete with a zero exit status.

The following example illustrates both how a variable assignment without a command name affects the current execution environment, and how an assignment with a command name only affects the execution environment of the command.

```

1632     $ x=red
1633     $ echo $x
1634     red
1635     $ export x
1636     $ sh -c 'echo $x'
1637     red
1638     $ x=blue sh -c 'echo $x'
1639     blue
1640     $ echo $x
1641     red

```

This next example illustrates that redirections without a command name are still performed.

```

1643     $ ls foo
1644     ls: foo: no such file or directory
1645     $ > foo
1646     $ ls foo
1647     foo

```

A command without a command name, but one that includes a command substitution, has an exit status of the last command substitution that the shell performed. For example:

```

1650     if      x=$(command)
1651     then    ...
1652     fi

```

An example of redirections without a command name being performed in a subshell shows that the here-document does not disrupt the standard input of the **while** loop:

```

1655     IFS=:
1656     while read a b
1657     do      echo $a
1658             <<-eof
1659             Hello
1660             eof
1661     done </etc/passwd

```

Some examples of commands without command names in AND-OR lists:

```
> foo || {
    echo "error: foo cannot be created" >&2
    exit 1
}

# set saved if /vmunix.save exists
test -f /vmunix.save && saved=1
```

Command substitution and redirections without command names both occur in subshells, but they are not necessarily the same ones. For example, in:

```
exec 3> file
var=$(echo foo >&3) 3>&1
```

it is unspecified whether **foo** will be echoed to the file or to standard output.

Command Search and Execution

If a simple command results in a command name and an optional list of arguments, the following actions will be performed.

1. If the command name does not contain any slashes, the first successful step in the following sequence will occur:

- a. If the command name matches the name of a special built-in utility, that special built-in utility will be invoked.
- b. If the command name matches the name of a function known to this shell, the function will be invoked as described in Section 2.9.5 on page 54. If the implementation has provided a standard utility in the form of a function, it will not be recognised at this point. It will be invoked in conjunction with the path search in step 1d.
- c. If the command name matches the name of a utility listed in the following table, that utility will be invoked.

<i>alias</i>	<i>false</i>	<i>jobs</i>	<i>true</i>
<i>bg</i>	<i>fc</i>	<i>kill</i>	<i>umask</i>
<i>cd</i>	<i>fg</i>	<i>newgrp</i>	<i>unalias</i>
<i>command</i>	<i>getopts</i>	<i>read</i>	<i>wait</i>

- d. Otherwise, the command will be searched for using the *PATH* environment variable as described in the **XBD** specification, **Chapter 6, Environment Variables**:

- i. If the search is successful:

- (a) If the system has implemented the utility as a regular built-in or as a shell function, it will be invoked at this point in the path search.
- (b) Otherwise, the shell will execute the utility in a separate utility environment (see Section 2.12 on page 63) with actions equivalent to calling the **XSH** specification *execve()* function with the *path* argument set to the pathname resulting from the search, *arg0* set to the command name, and the remaining arguments set to the operands, if any.

If the *execve()* function fails due to an error equivalent to the **XSH** specification error [ENOEXEC], the shell will execute a command equivalent to having a shell invoked with the command name as its first operand, along with any

1706 remaining arguments passed along. If the executable file is not a text file, the
 1707 shell may bypass this command execution, write an error message, and return
 1708 an exit status of 126.

1709 Once a utility has been searched for and found (either as a result of this specific
 1710 search or as part of an unspecified shell startup activity), an implementation
 1711 may remember its location and need not search for the utility again unless the
 1712 *PATH* variable has been the subject of an assignment. If the remembered
 1713 location fails for a subsequent invocation, the shell will repeat the search to find
 1714 the new location for the utility, if any.

1715 ii. If the search is unsuccessful, the command will fail with an exit status of 127
 1716 and the shell will write an error message.

1717 2. If the command name contains at least one slash, the shell will execute the utility in a
 1718 separate utility environment with actions equivalent to calling the **XSH** specification
 1719 *execve()* function with the *path* and *arg0* arguments set to the command name, and the
 1720 remaining arguments set to the operands, if any.

1721 If the *execve()* function fails due to an error equivalent to the **XSH** specification error
 1722 [ENOEXEC], the shell will execute a command equivalent to having a shell invoked with
 1723 the command name as its first operand, along with any remaining arguments passed
 1724 along. If the executable file is not a text file, the shell may bypass this command execution,
 1725 write an error message and return an exit status of 126.

1726 This description requires that the shell can execute shell scripts directly, even if the underlying
 1727 system does not support the common *#!* interpreter convention. That is, if file **foo** contains shell
 1728 commands and is executable, the following will execute **foo**:

1729 `./foo`

1730 The sequence selected for the ISO/IEC 9945-2: 1993 standard acknowledges that special built-ins
 1731 cannot be overridden, but gives the programmer full control over which versions of other
 1732 utilities are executed. It provides a means of suppressing function lookup (via the *command*
 1733 utility) for the user's own functions and ensures that any regular built-ins or functions provided
 1734 by the implementation are under the control of the path search. The mechanisms for associating
 1735 built-ins or functions with executable files in the path are not specified by this specification, but
 1736 the wording requires that if either is implemented, the application will not be able to distinguish
 1737 a function or built-in from an executable (other than in terms of performance, presumably). The
 1738 implementation will ensure that all effects specified by this specification resulting from the
 1739 invocation of the regular built-in or function (interaction with the environment, variables, traps,
 1740 and so on) are identical to those resulting from the invocation of an executable file.

1741 **Example:** Consider three versions of the *ls* utility:

- 1742 • The application includes a shell function named *ls*.
- 1743 • The user writes a utility named *ls* and puts it in **/fred/bin**.
- 1744 • The example implementation provides *ls* as a regular shell built-in that will be invoked
 1745 (either by the shell or directly by *exec*) when the path search reaches the directory **/posix/bin**.

1746 If PATH=/posix/bin, various invocations yield different versions of *ls*:

Invocation	Version of <i>ls</i>
1749 <i>ls</i> (from within application script)	(1) function
1750 <i>command ls</i> (from within application script)	(3) built-in
1751 <i>ls</i> (from within makefile called by application)	(3) built-in
1752 <i>system("ls")</i>	(3) built-in
1753 <i>PATH="/fred/bin:\$PATH" ls</i>	(2) user's version

1754 2.9.2 Pipelines

1755 A *pipeline* is a sequence of one or more commands separated by the control operator "|". The
 1756 standard output of all but the last command will be connected to the standard input of the next
 1757 command.

1758 The format for a pipeline is:

```
1759      [!] command1 [ | command2 ...]
```

1760 The standard output of *command1* will be connected to the standard input of *command2*. The
 1761 standard input, standard output or both of a command will be considered to be assigned by the
 1762 pipeline before any redirection specified by redirection operators that are part of the command
 1763 (see Section 2.7 on page 40).

1764 If the pipeline is not in the background (see Section 2.9.3 on page 50), the shell will wait for the
 1765 last command specified in the pipeline to complete, and may also wait for all commands to
 1766 complete.

1767 Exit Status

1768 If the reserved word "!" does not precede the pipeline, the exit status will be the exit status of the
 1769 last command specified in the pipeline. Otherwise, the exit status is the logical NOT of the exit
 1770 status of the last command. That is, if the last command returns zero, the exit status will be 1; if
 1771 the last command returns greater than zero, the exit status will be zero.

1772 Because pipeline assignment of standard input or standard output or both takes place before
 1773 redirection, it can be modified by redirection. For example:

```
1774      $ command1 2>&1 | command2
```

1775 sends both the standard output and standard error of *command1* to the standard input of
 1776 *command2*.

1777 The reserved word ! allows more flexible testing using AND and OR lists.

1778 2.9.3 Lists

1779 An *AND-OR-list* is a sequence of one or more pipelines separated by the operators:

```
1780      &&      | |
```

1781 A *list* is a sequence of one or more AND-OR-lists separated by the operators:

```
1782      ;      &
```

1783 and optionally terminated by:

```
1784      ;      &      <newline>
```

1785 The operators `&&` and `||` have equal precedence and will be evaluated from beginning to end.
 1786 For example, both of the following commands write solely **bar** to standard output:

```
1787     false && echo foo || echo bar
1788     true  || echo foo && echo bar
```

1789 A ";" or newline character terminator will cause the preceding AND-OR-list to be executed
 1790 sequentially; an "&" will cause asynchronous execution of the preceding AND-OR-list.

1791 The term *compound-list* is derived from the grammar in Section 2.10 on page 56; it is equivalent to
 1792 a sequence of *lists*, separated by newline characters, that can be preceded or followed by an
 1793 arbitrary number of newline characters.

1794 The following is an example that illustrates newline characters in compound-lists:

```
1795     while
1796     # a couple of newlines
1797
1798     # a list
1799     date && who || ls; cat file
1800     # a couple of newlines
1801
1802     # another list
1803     wc file > output & true
1804
1805     do
1806     # 2 lists
1807     ls
1808     cat file
1809
1810     done
```

1807 Asynchronous Lists

1808 If a command is terminated by the control operator ampersand (`&`), the shell will execute the
 1809 command asynchronously in a subshell. This means that the shell does not wait for the
 1810 command to finish before executing the next command.

1811 The format for running a command in the background is:

```
1812     command1 & [command2 & ...]
```

1813 The standard input for an asynchronous list, before any explicit redirections are performed, will
 1814 be considered to be assigned to a file that has the same properties as `/dev/null`. If it is an
 1815 interactive shell, this need not happen. In all cases, explicit redirection of standard input will
 1816 override this activity.

1817 Since the connection of the input to the equivalent of `/dev/null` is considered to occur before
 1818 redirections, the following script would produce no output:

```
1819     exec < /etc/passwd
1820     cat <&0 &
1821     wait
```

When an element of an asynchronous list (the portion of the list ended by an ampersand, such as *command1*, above) is started by the shell, the process ID of the last command in the asynchronous list element will become known in the current shell execution environment; see Section 2.12 on page 63. This process ID will remain known until:

1. The command terminates and the application waits for the process ID.
2. Another asynchronous list invoked before *\$!* (corresponding to the previous asynchronous list) is expanded in the current execution environment.

The implementation need not retain more than the {CHILD_MAX} most recent entries in its list of known process IDs in the current shell execution environment.

Exit Status: The exit status of an asynchronous list is zero.

Sequential Lists

Commands that are separated by a semicolon (;) will be executed sequentially.

The format for executing commands sequentially is:

```
command1 [ ; command2 ] ...
```

Each command will be expanded and executed in the order specified.

Exit Status: The exit status of a sequential list will be the exit status of the last command in the list.

AND Lists

The control operator && denotes an AND list. The format is:

```
command1 [ && command2 ] ...
```

First *command1* will be executed. If its exit status is zero, *command2* will be executed, and so on until a command has a non-zero exit status or there are no more commands left to execute. The commands will be expanded only if they are executed.

Exit Status: The exit status of an AND list will be the exit status of the last command that is executed in the list.

OR Lists

The control operator || denotes an OR List. The format is:

```
command1 [ || command2 ] ...
```

First, *command1* will be executed. If its exit status is non-zero, *command2* will be executed, and so on until a command has a zero exit status or there are no more commands left to execute.

Exit Status: The exit status of an OR list will be the exit status of the last command that is executed in the list.

1854 **2.9.4 Compound Commands**

1855 The shell has several programming constructs that are *compound commands*, which provide
 1856 control flow for commands. Each of these compound commands has a reserved word or control
 1857 operator at the beginning, and a corresponding terminator reserved word or operator at the end.
 1858 In addition, each can be followed by redirections on the same line as the terminator. Each
 1859 redirection will apply to all the commands within the compound command that do not explicitly
 1860 override that redirection.

1861 **Grouping Commands**

1862 The format for grouping commands is as follows:

1863 (*compound-list*) Execute *compound-list* in a subshell environment; see Section 2.12 on page
 1864 63. Variable assignments and built-in commands that affect the
 1865 environment will not remain in effect after the list finishes.

1866 { *compound-list*; } Execute *compound-list* in the current process environment. The semicolon
 1867 shown here is an example of a control operator delimiting the "}" reserved
 1868 word. Other delimiters are possible, as shown in Section 2.10 on page 56;
 1869 a newline character is frequently used.

1870 **Exit Status:** The exit status of a grouping command will be the exit status of *list*.

1871 **For Loop**

1872 The **for** loop will execute a sequence of commands for each member in a list of *items*. The **for**
 1873 loop requires that the reserved words **do** and **done** be used to delimit the sequence of
 1874 commands.

1875 The format for the **for** loop is as follows:

```
1876     for name [ in word ... ]
1877     do
1878         compound-list
1879     done
```

1880 First, the list of words following **in** will be expanded to generate a list of items. Then, the
 1881 variable *name* will be set to each item, in turn, and the *compound-list* executed each time. If no
 1882 items result from the expansion, the *compound-list* will not be executed. Omitting:

1883 in word...

1884 is equivalent to:

1885 in "\$@"

1886 The format is shown with generous usage of newline characters. See the grammar in Section
 1887 2.10 on page 56 for a precise description of where newline characters and semicolons can be
 1888 interchanged.

1889 **Exit Status:** The exit status of a **for** command will be the exit status of the last command that
 1890 executes. If there are no items, the exit status will be zero.

Case Conditional Construct

The conditional construct **case** will execute the *compound-list* corresponding to the first one of several *patterns* (see Section 2.13 on page 64) that is matched by the string resulting from the tilde expansion, parameter expansion, command substitution, and arithmetic expansion and quote removal of the given word. The reserved word **in** will denote the beginning of the patterns to be matched. Multiple patterns with the same *compound-list* are delimited by the "|" symbol. The control operator ")" terminates a list of patterns corresponding to a given action. The *compound-list* for each list of patterns is terminated with ;;. The **case** construct terminates with the reserved word **esac** (**case** reversed).

The format for the **case** construct is as follows:

```
case word in
    [ ( ]pattern1)          compound-list;;
    [ ( ]pattern2|pattern3) compound-list;;
    ...
esac
```

The ;; is optional for the last *compound-list*.

In order from the beginning to the end of the **case** statement, each *pattern* that labels a *compound-list* is subjected to tilde expansion, parameter expansion, command substitution and arithmetic expansion, and the result of these expansions is compared against the expansion of *word*, according to the rules described in Section 2.13 on page 64 (which also describes the effect of quoting parts of the pattern). After the first match, no more patterns are expanded, and the *compound-list* is executed. The order of expansion and comparison of multiple *patterns* that label a *compound-list* statement is unspecified.

Exit Status: The exit status of **case** is zero if no patterns are matched. Otherwise, the exit status will be the exit status of the last command executed in the *compound-list*.

The pattern *, given as the last pattern in a **case** construct, is equivalent to the default case in a C-language **switch** statement.

The grammar shows that reserved words can be used as patterns, even if one is the first word on a line. Obviously, the reserved word **esac** cannot be used in this manner.

If Conditional Construct

The **if** command will execute a *compound-list* and use its exit status to determine whether to execute another *compound-list*.

The format for the **if** construct is as follows:

```
if compound-list
then
    compound-list
[elif compound-list
then
    compound-list] ...
[else
    compound-list]
fi
```

The **if** *compound-list* is executed; if its exit status is zero, the **then** *compound-list* is executed and the command will complete. Otherwise, each **elif** *compound-list* is executed, in turn, and if its exit status is zero, the **then** *compound-list* is executed and the command will complete.

1936 Otherwise, the **else** *compound-list* is executed.

1937 **Exit Status:** The exit status of the **if** command will be the exit status of the *then* or **else**
 1938 *compound-list* that was executed, or zero, if none was executed.

1939 **While Loop**

1940 The **while** loop continuously will execute one *compound-list* as long as another *compound-list* has
 1941 a zero exit status.

1942 The format of the **while** loop is as follows:

```
1943     while compound-list-1
1944     do
1945         compound-list-2
1946     done
```

1947 The *compound-list-1* will be executed, and if it has a non-zero exit status, the **while** command will
 1948 complete. Otherwise, the *compound-list-2* will be executed, and the process will repeat.

1949 **Exit Status:** The exit status of the **while** loop will be the exit status of the last *compound-list-2*
 1950 executed, or zero if none was executed.

1951 **Until Loop**

1952 The **until** loop continuously will execute one *compound-list* as long as another *compound-list* has a
 1953 non-zero exit status.

1954 The format of the **until** loop is as follows:

```
1955     until compound-list-1
1956     do
1957         compound-list-2
1958     done
```

1959 The *compound-list-1* will be executed, and if it has a zero exit status, the **until** command will
 1960 complete. Otherwise, the *compound-list-2* will be executed, and the process will repeat.

1961 **Exit Status:** The exit status of the **until** loop will be the exit status of the last *compound-list-2*
 1962 executed, or zero if none was executed.

1963 **2.9.5 Function Definition Command**

1964 A function is a user-defined name that is used as a simple command to call a compound
 1965 command with new positional parameters. A function is defined with a *function definition*
 1966 *command*.

1967 The format of a function definition command is as follows:

```
1968     fname() compound-command[io-redirect ...]
```

1969 The function is named *fname*; it must be a name (see **name** in the XBD specification, **Chapter 2,**
 1970 **Glossary**). An implementation may allow other characters in a function name as an extension.
 1971 The implementation will maintain separate name spaces for functions and variables.

1972 The () in the function definition command consists of two operators. Therefore, intermixing
 1973 blank characters with the *fname*, "(" and ")" is allowed, but unnecessary.

1974 The argument *compound-command* represents a compound command, as described in Section
 1975 2.9.4 on page 52.

1976 When the function is declared, none of the expansions in Section 2.6 on page 31 will be
 1977 performed on the text in *compound-command* or *io-redirect*; all expansions will be performed as
 1978 normal each time the function is called. Similarly, the optional *io-redirect* redirections and any
 1979 variable assignments within *compound-command* will be performed during the execution of the
 1980 function itself, not the function definition. See Section 2.8.1 on page 44 for the consequences of
 1981 failures of these operations on interactive and non-interactive shells.

1982 When a function is executed, it will have the syntax-error and variable-assignment properties
 1983 described for special built-in utilities in the enumerated list at the beginning of Section 2.14 on
 1984 page 67.

1985 The *compound-command* will be executed whenever the function name is specified as the name of
 1986 a simple command (see **Command Search and Execution** on page 47). The operands to the
 1987 command temporarily will become the positional parameters during the execution of the
 1988 *compound-command*; the special parameter "#" will also be changed to reflect the number of
 1989 operands. The special parameter 0 will be unchanged. When the function completes, the values
 1990 of the positional parameters and the special parameter "#" will be restored to the values they had
 1991 before the function was executed. If the special built-in *return* is executed in the *compound-*
 1992 *command*, the function will complete and execution will resume with the next command after the
 1993 function call.

1994 An example of how a function definition can be used wherever a simple command is allowed:

```
1995     # If variable i is equal to "yes",
1996     # define function foo to be ls -l
1997     #
1998     [ "$i" = yes ] && foo() {
1999         ls -l
2000     }
```

2001 **Exit Status**

2002 The exit status of a function definition will be zero if the function was declared successfully;
 2003 otherwise, it will be greater than zero. The exit status of a function invocation will be the exit
 2004 status of the last command executed by the function.

2005 **2.10 Shell Grammar**

2006 The following grammar defines the Shell Command Language. This formal syntax takes
 2007 precedence over the preceding text syntax description.

2008 **2.10.1 Shell Grammar Lexical Conventions**

2009 The input language to the shell must be first recognised at the character level. The resulting
 2010 tokens will be classified by their immediate context according to the following rules (applied in
 2011 order). These rules are used to determine what a “token” that is subject to parsing at the token
 2012 level is. The rules for token recognition in Section 2.3 on page 23 will apply.

- 2013 1. A newline character will be returned as the token identifier **NEWLINE**.
- 2014 2. If the token is an operator, the token identifier for that operator will result.
- 2015 3. If the string consists solely of digits and the delimiter character is one of < or >, the token
 2016 identifier **IO_NUMBER** will be returned.
- 2017 4. Otherwise, the token identifier **TOKEN** will result.

2018 Further distinction on **TOKEN** is context-dependent. It may be that the same **TOKEN** yields
 2019 **WORD**, a **NAME**, an **ASSIGNMENT**, or one of the reserved words below, dependent upon the
 2020 context. Some of the productions in the grammar below are annotated with a rule number from
 2021 the following list. When a **TOKEN** is seen where one of those annotated productions could be
 2022 used to reduce the symbol, the applicable rule will be applied to convert the token identifier type
 2023 of the **TOKEN** to a token identifier acceptable at that point in the grammar. The reduction will
 2024 then proceed based upon the token identifier type yielded by the rule applied. When more than
 2025 one rule applies, the highest numbered rule will apply (which in turn may refer to another rule).
 2026 (Note that except in rule 7, the presence of an = in the token has no effect.)

2027 The **WORD** tokens will have the word expansion rules applied to them immediately before the
 2028 associated command is executed, not at the time the command is parsed.

2029 **2.10.2 Shell Grammar Rules**

- 2030 1. [Command Name]
 2031 When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word
 2032 will result. Otherwise, the token **WORD** will be returned. Also, if the parser is in any state
 2033 where only a reserved word could be the next correct token, proceed as above. This rule
 2034 applies rather narrowly: when a compound list is terminated by some clear delimiter
 2035 (such as the closing **fi** of an inner **if_clause**) then it would apply; where the compound list
 2036 might continue (as in after a **;**), rule 7a (and consequently the first sentence of this rule)
 2037 would apply. In many instances the two conditions are identical, but this part of this rule
 2038 does not give licence to treating a **WORD** as a reserved word unless it is in a place where a
 2039 reserved word must appear.

2040 **Note:** Because at this point quote marks are retained in the token, quoted strings cannot
 2041 be recognised as reserved words. This rule also implies that reserved words will
 2042 not be recognised except in certain positions in the input, such as after a newline
 2043 character or semicolon; the grammar presumes that if the reserved word is
 2044 intended, it will be properly delimited by the user, and does not attempt to reflect
 2045 that requirement directly. Also note that line joining is done before tokenisation,
 2046 as described in Section 2.2.1 on page 20, so escaped newlines are already removed
 2047 at this point.

- 2048 Rule 1 is not directly referenced in the grammar, but is referred to by other rules, or applies
2049 globally.
- 2050 2. [Redirection to or from filename]
2051 The expansions specified in Section 2.7 on page 40 will occur. As specified there, exactly
2052 one field can result (or the result is unspecified), and there are additional requirements on
2053 pathname expansion.
- 2054 3. [Redirection from here-document]
2055 Quote removal will be applied to the word to determine the delimiter that will be used to
2056 find the end of the here-document that begins after the next newline character.
- 2057 4. [Case statement termination]
2058 When the **TOKEN** is exactly the reserved word **Esac**, the token identifier for **Esac** will
2059 result. Otherwise, the token **WORD** will be returned.
- 2060 5. [NAME in **for**]
2061 When the **TOKEN** meets the requirements for a name (see **name** in the **XBD** specification,
2062 **Chapter 2, Glossary**), the token identifier **NAME** will result. Otherwise, the token **WORD**
2063 will be returned.
- 2064 6. [Third word of **for** and **case**]
2065 When the **TOKEN** is exactly the reserved word **In**, the token identifier for **In** will result.
2066 Otherwise, the token **WORD** will be returned. (As indicated in the grammar, a **linebreak**
2067 precedes the token **In**. If newline characters are present at the indicated location, it is the
2068 token after them that is treated in this fashion.)
- 2069 7. [Assignment preceding command name]
- 2070 a. [When the first word]
2071 If the **TOKEN** does not contain the character "=", rule 1 will be applied. Otherwise, |
2072 7b will be applied.
- 2073 b. [Not the first word]
2074 If the **TOKEN** contains the equal sign character:
- 2075 — If it begins with =, the token **WORD** will be returned.
- 2076 — If all the characters preceding = form a valid name (see **name** in the **XSH**
2077 specification), the token **ASSIGNMENT_WORD** will be returned. (Quoted
2078 characters cannot participate in forming a valid name.)
- 2079 — Otherwise, it is unspecified whether it is **ASSIGNMENT_WORD** or **WORD** that
2080 is returned.
- 2081 Assignment to the **NAME** will occur as specified in Section 2.9.1 on page 45.
- 2082 8. [NAME in function]
2083 When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word
2084 will result. Otherwise, when the **TOKEN** meets the requirements for a name (see **Name**),
2085 the token identifier **NAME** will result. Otherwise, rule 7 will apply.
- 2086 9. [Body of function]
2087 Word expansion and assignment will never occur, even when required by the rules above,
2088 when this rule is being parsed. Each **TOKEN** that might either be expanded or have
2089 assignment applied to it will instead be returned as a single **WORD** consisting only of
2090 characters that are exactly the token described in Section 2.3 on page 23.

```

2091      /* -----
2092      The grammar symbols
2093      ----- */

2094      %token  WORD
2095      %token  ASSIGNMENT_WORD
2096      %token  NAME
2097      %token  NEWLINE
2098      %token  IO_NUMBER

2099      /* The following are the operators mentioned above. */

2100      %token  AND_IF      OR_IF      DSEMI
2101      /*      '&&'      '|'      ';'      */

2102      %token  DLESS      DGREAT      LESSAND      GREATAND      LESSGREAT      DLESSDASH
2103      /*      '<<'      '>>'      '<&'      '>&'      '<>'      '<<-'      */

2104      %token  CLOBBER
2105      /*      '>|'      */

2106      /* The following are the reserved words. */

2107      %token  If      Then      Else      Elif      Fi      Do      Done
2108      /*      'if'      'then'      'else'      'elif'      'fi'      'do'      'done'      */

2109      %token  Case      Esac      While      Until      For
2110      /*      'case'      'esac'      'while'      'until'      'for'      */

2111      /* These are reserved words, not operator tokens, and are
2112         recognised when reserved words are recognised. */

2113      %token  Lbrace      Rbrace      Bang
2114      /*      '{'      '}'      '!'      */

2115      %token  In
2116      /*      'in'      */

2117      /* -----
2118      The Grammar
2119      ----- */

2120      %start  complete_command
2121      %%
2122      complete_command : list separator
2123                      | list
2124                      ;
2125      list             : list separator_op and_or
2126                      |                               and_or
2127                      ;
2128      and_or           :                               pipeline
2129                      | and_or AND_IF linebreak pipeline
2130                      | and_or OR_IF  linebreak pipeline
2131                      ;
2132      pipeline         :      pipe_sequence
2133                      | Bang pipe_sequence
2134                      ;
2135      pipe_sequence    :                               command
2136                      | pipe_sequence '|' linebreak command

```

```

2137                                     ;
2138     command                         : simple_command
2139                                     | compound_command
2140                                     | compound_command redirect_list
2141                                     | function_definition
2142                                     ;
2143     compound_command                 : brace_group
2144                                     | subshell
2145                                     | for_clause
2146                                     | case_clause
2147                                     | if_clause
2148                                     | while_clause
2149                                     | until_clause
2150                                     ;
2151     subshell                         : '(' compound_list ')'
2152                                     ;
2153     compound_list                    : term
2154                                     | newline_list term
2155                                     | term separator
2156                                     | newline_list term separator
2157                                     ;
2158     term                             : term separator and_or
2159                                     | and_or
2160                                     ;
2161     for_clause                       : For name linebreak do_group
2162                                     | For name linebreak in wordlist sequential_sep do_group
2163                                     ;
2164     name                             : NAME /* Apply rule 5 */
2165                                     ;
2166     in                               : In /* Apply rule 6 */
2167                                     ;
2168     wordlist                         : wordlist WORD
2169                                     | WORD
2170                                     ;
2171     case_clause                      : Case WORD linebreak in linebreak case_list Esac
2172                                     | Case WORD linebreak in linebreak Esac
2173                                     ;
2174     case_list                        : case_list case_item
2175                                     | case_item
2176                                     ;
2177     case_item                        : pattern ')' linebreak DSEMI linebreak
2178                                     | pattern ')' compound_list DSEMI linebreak
2179                                     | '(' pattern ')' linebreak DSEMI linebreak
2180                                     | '(' pattern ')' compound_list DSEMI linebreak
2181                                     ;
2182     pattern                          : WORD /* Apply rule 4 */
2183                                     | pattern '|' WORD /* Do not apply rule (4) */
2184                                     ;
2185     if_clause                        : If compound_list Then compound_list else_part Fi
2186                                     | If compound_list Then compound_list Fi
2187                                     ;
2188     else_part                        : Elif compound_list Then else_part

```

```

2189             | Else compound_list
2190             ;
2191 while_clause   : While compound_list do_group
2192             ;
2193 until_clause   : Until compound_list do_group
2194             ;
2195 function_definition : fname '(' ')' linebreak function_body
2196             ;
2197 function_body   : compound_command /* Apply rule 9 */
2198             | compound_command redirect_list /* Apply rule 9 */
2199             ;
2200 fname          : NAME /* Apply rule 8 */
2201             ;
2202 brace_group     : Lbrace compound_list Rbrace
2203             ;
2204 do_group        : Do compound_list Done
2205             ;
2206 simple_command  : cmd_prefix cmd_word cmd_suffix
2207             | cmd_prefix cmd_word
2208             | cmd_prefix
2209             | cmd_name cmd_suffix
2210             | cmd_name
2211             ;
2212 cmd_name        : WORD /* Apply rule 7a */
2213             ;
2214 cmd_word        : WORD /* Apply rule 7b */
2215             ;
2216 cmd_prefix      : io_redirect
2217             | cmd_prefix io_redirect
2218             | ASSIGNMENT_WORD
2219             | cmd_prefix ASSIGNMENT_WORD
2220             ;
2221 cmd_suffix      : io_redirect
2222             | cmd_suffix io_redirect
2223             | WORD
2224             | cmd_suffix WORD
2225             ;
2226 redirect_list   : io_redirect
2227             | redirect_list io_redirect
2228             ;
2229 io_redirect     : io_file
2230             | IO_NUMBER io_file
2231             | io_here
2232             | IO_NUMBER io_here
2233             ;
2234 io_file         : '<' filename
2235             | LESSAND filename
2236             | '>' filename
2237             | GREATAND filename
2238             | DGREAT filename
2239             | LESSGREAT filename
2240             | CLOBBER filename

```

```

2241      ;
2242      filename      : WORD                      /* Apply rule 2 */
2243      ;
2244      io_here       : DLESS      here_end
2245      |              DLESSDASH here_end
2246      ;
2247      here_end      : WORD                      /* Apply rule 3 */
2248      ;
2249      newline_list  : NEWLINE
2250      | newline_list NEWLINE
2251      ;
2252      linebreak     : newline_list
2253      | /* empty */
2254      ;
2255      separator_op   : '&'
2256      | ';'
2257      ;
2258      separator     : separator_op linebreak
2259      | newline_list
2260      ;
2261      sequential_sep : ';' linebreak
2262      | newline_list
2263      ;

```

2264 There are several subtle aspects of this grammar where conventional usage implies rules about
 2265 the grammar that in fact are not true.

2266 For **compound_list**, only the forms that end in a **separator** allow a reserved word to be
 2267 recognised, so usually only a **separator** can be used where a compound list precedes a reserved
 2268 word (such as **Then**, **Else**, **Do** and **Rbrace**). Explicitly requiring a separator would disallow such
 2269 valid (if rare) statements as:

```

2270      if (false) then (echo x) else (echo y) fi

```

2271 See the Note under special grammar rule 1.

2272 Note that the bodies of here-documents are handled by token recognition (see Section 2.3 on
 2273 page 23) and do not appear in the grammar directly. (However, the here-document I/O
 2274 redirection operator is handled as part of the grammar.)

2275 The start symbol of the grammar (**complete_command**) represents either input from the
 2276 command line or a shell script. It is repeatedly applied by the interpreter to its input, and
 2277 represents a single chunk of that input as seen by the interpreter.

2.11 Signals and Error Handling

When a command is in an asynchronous list, the shell will prevent SIGQUIT and SIGINT signals from the keyboard from interrupting the command. Otherwise, signals will have the values inherited by the shell from its parent (see also the *trap* special built-in).

When a signal for which a trap has been set is received while the shell is waiting for the completion of a utility executing a foreground command, the trap associated with that signal will not be executed until after the foreground command has completed. When the shell is waiting, by means of the *wait* utility, for asynchronous commands to complete, the reception of a signal for which a trap has been set will cause the *wait* utility to return immediately with an exit status >128, immediately after which the trap associated with that signal will be taken.

If multiple signals are pending for the shell for which there are associated trap actions, the order of execution of trap actions is unspecified.

2.12 Shell Execution Environment

A shell execution environment consists of the following:

- open files inherited upon invocation of the shell, plus open files controlled by *exec*
- working directory as set by *cd*
- file creation mask set by *umask*
- current traps set by *trap*
- shell parameters that are set by variable assignment (see the *set* special built-in) or from the **XSH** specification environment inherited by the shell when it begins (see the *export* special built-in)
- shell functions (see Section 2.9.5 on page 54)
- options turned on at invocation or by *set*
- process IDs of the last commands in asynchronous lists known to this shell environment; see Section 2.9.3 on page 50
- shell aliases (see Section 2.3.1 on page 24).

Utilities other than the special built-ins (see Section 2.14 on page 67) will be invoked in a separate environment that consists of the following. The initial value of these objects will be the same as that for the parent shell, except as noted below.

- open files inherited on invocation of the shell, open files controlled by the *exec* special built-in plus any modifications and additions specified by any redirections to the utility
- current working directory
- file creation mask
- if the utility is a shell script, traps caught by the shell will be set to the default values and traps ignored by the shell will be set to be ignored by the utility; if the utility is not a shell script, the trap actions (default or ignore) will be mapped into the appropriate signal handling actions for the utility
- variables with the *export* attribute, along with those explicitly exported for the duration of the command, will be passed to the utility as **XSH** specification environment variables.

The environment of the shell process will not be changed by the utility unless explicitly specified by the utility description (for example, *cd* and *umask*).

A subshell environment will be created as a duplicate of the shell environment, except that signal traps set by that shell environment will be set to the default values. Changes made to the subshell environment will not affect the shell environment. Command substitution, commands that are grouped with parentheses and asynchronous lists will be executed in a subshell environment. Additionally, each command of a multi-command pipeline is in a subshell environment; as an extension, however, any or all commands in a pipeline may be executed in the current environment. All other commands will be executed in the current shell environment.

Some systems have implemented the last stage of a pipeline in the current environment so that commands such as:

```
command | read foo
```

set variable **foo** in the current environment. This extension is allowed, but not required; therefore, a shell programmer should consider a pipeline to be in a subshell environment, but not depend on it.

2.13 Pattern Matching Notation

The pattern matching notation described in this section is used to specify patterns for matching strings in the shell. Historically, pattern matching notation is related to, but slightly different from, the regular expression notation described in the **XBD** specification, **Chapter 7, Regular Expressions**. For this reason, the description of the rules for this pattern matching notation are based on the description of regular expression notation.

2.13.1 Patterns Matching a Single Character

The following *patterns matching a single character* match a single character: *ordinary characters*, *special pattern characters* and *pattern bracket expressions*. The pattern bracket expression will also match a single collating element.

An ordinary character is a pattern that matches itself. It can be any character in the supported character set except for NUL, those special shell characters in Section 2.2 on page 20 that require quoting, and the following three special pattern characters. Matching is based on the bit pattern used for encoding the character, not on the graphic representation of the character. If any character (ordinary, shell special or pattern special) is quoted, that pattern will match the character itself. The shell special characters always require quoting.

When unquoted and outside a bracket expression, the following three characters will have special meaning in the specification of patterns:

- ? A question-mark is a pattern that will match any character.
- * An asterisk is a pattern that will match multiple characters, as described in Section 2.13.2 on page 65.
- [The open bracket will introduce a pattern bracket expression.

The description of basic regular expression bracket expressions in the **XBD** specification, **Section 7.3.5, RE Bracket Expression** also applies to the pattern bracket expression, except that the exclamation-mark character (!) replaces the circumflex character (^) in its role in a *non-matching list* in the regular expression notation. A bracket expression starting with an unquoted circumflex character produces unspecified results.

The restriction on a circumflex in a bracket expression is to allow implementations that support pattern matching using the circumflex as the negation character in addition to the exclamation-mark. A portable application must use something like `[^!]` to match either character.

When pattern matching is used where shell quote removal is not performed (such as in the argument to the *find -name* primary when *find* is being called using one of the **XSH** specification *exec* functions, or in the *pattern* argument to the *fnmatch()* function), special characters can be escaped to remove their special meaning by preceding them with a backslash character. This escaping backslash will be discarded. The sequence `\\` represents one literal backslash. All of the requirements and effects of quoting on ordinary, shell special and special pattern characters will apply to escaping in this context.

Both quoting and escaping are described here because pattern matching must work in three separate circumstances:

- Calling directly upon the shell, such as in pathname expansion or in a **case** statement. All of the following will match the string or file **abc**:

abc "abc" a"b"c a\bc a[b]c a["b"]c a[\b]c a["\b"]c a?c a*c

2374 The following will not:

2375 `"a?c" a*c a\[b]c`

2376 • Calling a utility or function without going through a shell, as described for *find* and the **XSH**
2377 specification function *fnmatch()*.

2378 • Calling utilities such as *find*, *cpio*, *tar* or *pax* through the shell command line. In this case,
2379 shell quote removal is performed before the utility sees the argument. For example, in:

2380 `find /bin -name "e\c[\h]o" -print`

2381 after quote removal, the backslashes are presented to *find* and it treats them as escape
2382 characters. Both precede ordinary characters, so the **c** and **h** represent themselves and **echo**
2383 would be found on many historical systems (that have it in **/bin**). To find a filename that
2384 contained shell special characters or pattern characters, both quoting and escaping are
2385 required, such as:

2386 `pax -r ... "*a\(\?"`

2387 to extract a filename ending with **a(?)**.

2388 Conforming applications are required to quote or escape the shell special characters (sometimes
2389 called metacharacters). If used without this protection, syntax errors can result or
2390 implementation extensions can be triggered. For example, the KornShell supports a series of
2391 extensions based on parentheses in patterns.

2392 2.13.2 Patterns Matching Multiple Characters

2393 The following rules are used to construct *patterns matching multiple characters* from *patterns*
2394 *matching a single character*:

- 2395 1. The asterisk (*) is a pattern that will match any string, including the null string.
- 2396 2. The concatenation of *patterns matching a single character* is a valid pattern that will match
2397 the concatenation of the single characters or collating elements matched by each of the
2398 concatenated patterns.
- 2399 3. The concatenation of one or more *patterns matching a single character* with one or more
2400 asterisks is a valid pattern. In such patterns, each asterisk will match a string of zero or
2401 more characters, matching the greatest possible number of characters that still allows the
2402 remainder of the pattern to match the string.

2403 Since each asterisk matches zero or more occurrences, the patterns **a*b** and **a**b** have identical
2404 functionality.

2405 Examples

2406 **a[bc]** matches the strings **ab** and **ac**.

2407 **a*d** matches the strings **ad**, **abd** and **abcd**, but not the string **abc**.

2408 **a*d*** matches the strings **ad**, **abcd**, **abcdef**, **aaaad** and **adddd**.

2409 ***a*d** matches the strings **ad**, **abcd**, **efabcd**, **aaaad** and **adddd**.

2410 2.13.3 Patterns Used for Filename Expansion

2411 The rules described so far in Section 2.13.1 on page 64 and Section 2.13.2 on page 65 are qualified
 2412 by the following rules that apply when pattern matching notation is used for filename
 2413 expansion.

2414 1. The slash character in a pathname must be explicitly matched by using one or more slashes
 2415 in the pattern; it cannot be matched by the asterisk or question-mark special characters or
 2416 by a bracket expression. Slashes in the pattern are identified before bracket expressions;
 2417 thus, a slash cannot be included in a pattern bracket expression used for filename
 2418 expansion. For example, the pattern `a[b/c]d` will not match such pathnames as **abd** or **a/d**.
 2419 It will only match a pathname of literally **a[b/c]d**.

2420 2. If a filename begins with a period (.) the period must be explicitly matched by using a
 2421 period as the first character of the pattern or immediately following a slash character. The
 2422 leading period will not be matched by:

- 2423 • the asterisk or question-mark special characters
- 2424 • a bracket expression containing a non-matching list, such as:

2425 `[!a]`

2426 a range expression, such as:

2427 `[%-0]`

2428 or a character class expression, such as:

2429 `[[:punct:]]`

2430 It is unspecified whether an explicit period in a bracket expression matching list, such as:

2431 `[.abc]`

2432 can match a leading period in a filename.

2433 3. Specified patterns are matched against existing filenames and pathnames, as appropriate.
 2434 Each component that contains a pattern character requires read permission in the directory
 2435 containing that component. Any component, except the last, that does not contain a
 2436 pattern character requires search permission. For example, given the pattern:

2437 `/foo/bar/x*/bam`

2438 search permission is needed for directories `/` and **foo**, search and read permissions are
 2439 needed for directory **bar**, and search permission is needed for each **x*** directory. If the
 2440 pattern matches any existing filenames or pathnames, the pattern will be replaced with
 2441 those filenames and pathnames, sorted according to the collating sequence in effect in the
 2442 current locale. If the pattern contains an invalid bracket expression or does not match any
 2443 existing filenames or pathnames, the pattern string is left unchanged.

2444 2.14 Special Built-in Utilities

2445 The following *special built-in* utilities will be supported in the shell command language. The
2446 output of each command, if any, will be written to standard output, subject to the normal
2447 redirection and piping possible with all commands.

2448 The term *built-in* implies that the shell can execute the utility directly and does not need to
2449 search for it. An implementation can choose to make any utility a built-in; however, the special
2450 built-in utilities described here differ from regular built-in utilities in two respects:

- 2451 • A syntax error in a special built-in utility may cause a shell executing that utility to abort,
2452 while a syntax error in a regular built-in utility will not cause a shell executing that utility to
2453 abort. (See Section 2.8.1 on page 44 for the consequences of errors on interactive and non-
2454 interactive shells.) If a special built-in utility encountering a syntax error does not abort the
2455 shell, its exit value will be non-zero.
- 2456 • Variable assignments specified with special built-in utilities will remain in effect after the
2457 built-in completes; this is not the case with a regular built-in or other utility.

2458 The special built-in utilities in this section need not be provided in a manner accessible via the
2459 **XSH** specification *exec* family of functions.

2460 Some of the special built-ins are described as conforming to the **XBD** specification, **Section 10.2,**
2461 **Utility Syntax Guidelines**. For those that are not, the requirement in Section 1.9 on page 11 that
2462 -- be recognised as a first argument to be discarded does not apply and a portable application
2463 must not use that argument.

2464 **2.14.1 break — Exit From for, while or until Loop**2465 **SYNOPSIS**2466 *break* [*n*]2467 **DESCRIPTION**

2468 Exit from the smallest enclosing **for**, **while** or **until** loop, if any; or from the *n*th enclosing loop if
2469 *n* is specified. The value of *n* is an unsigned decimal integer greater than or equal to 1. The
2470 default is equivalent to *n*=1. If *n* is greater than the number of enclosing loops, the last enclosing
2471 loop is exited from. Execution will continue with the command immediately following the loop.

2472 **EXAMPLES**

```
2473 for i in *
2474 do
2475     if test -d "$i"
2476     then break
2477     fi
2478 done
```

2479 **EXIT STATUS**

2480 0 Successful completion.
2481 >0 The *n* value was not an unsigned decimal integer greater than or equal to 1.

2482 **2.14.2 colon — Null Utility**2483 **SYNOPSIS**2484 : [*argument* ...]2485 **DESCRIPTION**

2486 This utility will only expand command *arguments*. It is used when a command is needed, as in
2487 the *then* condition of an **if** command, but nothing is to be done by the command.

2488 **EXAMPLES**

```
2489 : ${X=abc}  
2490 if      false  
2491 then   :  
2492 else   echo $X  
2493 fi  
2494 abc
```

2495 As with any of the special built-ins, the null utility can also have variable assignments and
2496 redirections associated with it, such as:

```
2497      x=y : > z
```

2498 which sets variable **x** to the value **y** (so that it persists after the null utility completes) and creates
2499 or truncates file **z**.

2500 **EXIT STATUS**

2501 Zero.

2502 **2.14.3 continue — Continue for, while or until Loop**2503 **SYNOPSIS**2504 *continue* [*n*]2505 **DESCRIPTION**

2506 The *continue* utility will return to the top of the smallest enclosing **for**, **while** or **until** loop, or to
2507 the top of the *n*th enclosing loop, if *n* is specified. This involves repeating the condition list of a
2508 **while** or **until** loop or performing the next assignment of a **for** loop, and reexecuting the loop if
2509 appropriate.

2510 The value of *n* is a decimal integer greater than or equal to 1. The default is equivalent to *n*=1. If
2511 *n* is greater than the number of enclosing loops, the last enclosing loop is used.

2512 **EXAMPLES**

```
2513 for i in *
2514 do
2515     if test -d "$i"
2516     then continue
2517     fi
2518 done
```

2519 **EXIT STATUS**

2520 0 Successful completion.
2521 >0 The *n* value was not an unsigned decimal integer greater than or equal to 1.

2522 2.14.4 dot — Execute Commands in Current Environment**2523 SYNOPSIS**

2524 . *file*

2525 DESCRIPTION

2526 The shell will execute commands from the *file* in the current environment.

2527 If *file* does not contain a slash, the shell will use the search path specified by *PATH* to find the
2528 directory containing *file*. Unlike normal command search, however, the file searched for by the
2529 *dot* utility need not be executable. If no readable file is found, a non-interactive shell will abort;
2530 an interactive shell will write a diagnostic message to standard error, but this condition will not
2531 be considered a syntax error.

2532 EXAMPLES

2533 cat foobar
2534 **foo=hello bar=world**
2535 . foobar
2536 echo \$foo \$bar
2537 **hello world**

2538 EXIT STATUS

2539 Returns the value of the last command executed, or a zero exit status if no command is executed.

2540 **2.14.5 eval — Construct Command by Concatenating Arguments**2541 **SYNOPSIS**2542 *eval* [*argument* . . .]2543 **DESCRIPTION**2544 The *eval* utility will construct a command by concatenating *arguments* together, separating each
2545 with a space character. The constructed command will be read and executed by the shell. |2546 **EXAMPLES**2547 *foo*=10 *x*=*foo* |2548 *y*=' '\$*x*2549 *echo* *\$y*2550 ***\$foo***2551 *eval* *y*=' '\$*x*2552 *echo* *\$y*2553 **10**2554 **EXIT STATUS**2555 If there are no *arguments*, or only null arguments, *eval* will return a zero exit status; otherwise, it
2556 will return the exit status of the command defined by the string of concatenated *arguments*
2557 separated by spaces. |

2558 **2.14.6 exec — Execute Commands and Open, Close or Copy File Descriptors**2559 **SYNOPSIS**2560 `exec [command [argument ...]]`2561 **DESCRIPTION**2562 The *exec* utility will open, close or copy file descriptors as specified by any redirections as part of
2563 the command.2564 If *exec* is specified without *command* or *arguments*, and any file descriptors with numbers > 2 are
2565 opened with associated redirection statements, it is unspecified whether those file descriptors
2566 remain open when the shell invokes another utility. Scripts concerned that child shells could
2567 misuse open file descriptors can always close them explicitly, as shown in one of the following
2568 examples.2569 If *exec* is specified with *command*, it will replace the shell with *command* without creating a new
2570 process. If *arguments* are specified, they are arguments to *command*. Redirection will affect the
2571 current shell execution environment.2572 **EXAMPLES**2573 Open **readfile** as file descriptor 3 for reading:2574 `exec 3< readfile`2575 Open **writefile** as file descriptor 4 for writing:2576 `exec 4> writefile`

2577 Make unit 5 a copy of unit 0:

2578 `exec 5<&0`

2579 Close file unit 3:

2580 `exec 3<&-`2581 Cat the file **maggie** by replacing the current shell with the *cat* utility:2582 `exec cat maggie`2583 **EXIT STATUS**2584 If *command* is specified, *exec* will not return to the shell; rather, the exit status of the process will
2585 be the exit status of the program implementing *command*, which overlaid the shell. If *command* is
2586 not found, the exit status will be 127. If *command* is found, but it is not an executable utility, the
2587 exit status will be 126. If a redirection error occurs (see Section 2.8.1 on page 44), the shell will
2588 exit with a value in the range 1–125. Otherwise, *exec* will return a zero exit status.

2589 **2.14.7 exit — Cause the Shell to Exit**2590 **SYNOPSIS**2591 *exit* [*n*]2592 **DESCRIPTION**

2593 The *exit* utility causes the shell to exit with the exit status specified by the unsigned decimal
2594 integer *n*. If *n* is specified, but its value is not between 0 and 255 inclusively, the exit status is
2595 undefined.

2596 As explained in other sections, certain exit status values have been reserved for special uses and
2597 should be used by applications only for those purposes:

2598 126 A command to be executed was found, but the file was not an executable utility.

2599 127 A command to be executed was not found.

2600 >128 A command was interrupted by a signal.

2601 A trap on **EXIT** will be executed before the shell terminates, except when the *exit* utility is
2602 invoked in that trap itself, in which case the shell will exit immediately.

2603 **EXAMPLES**

2604 Exit with a *true* value:

2605 *exit* 0

2606 Exit with a *false* value:

2607 *exit* 1

2608 **EXIT STATUS**

2609 The exit status will be *n*, if specified. Otherwise, the value will be the exit value of the last
2610 command executed, or zero if no command was executed. When *exit* is executed in a trap
2611 action, the last command is considered to be the command that executed immediately preceding
2612 the trap action.

2613 **2.14.8 export — Set Export Attribute for Variables**2614 **SYNOPSIS**

2615 *export* *name*[=*word*]...
 2616 *export* -p

2617 **DESCRIPTION**

2618 The shell will give the export attribute to the variables corresponding to the specified *names*,
 2619 which will cause them to be in the environment of subsequently executed commands.

2620 The *export* special built-in supports the **XBD** specification, **Section 10.2, Utility Syntax**
 2621 **Guidelines**.

2622 When -p is specified, *export* will write to the standard output the names and values of all
 2623 exported variables, in the following format:

2624 "export %s=%s\n", <name>, <value>

2625 The -p option allows portable access to the values that can be saved and then later restored
 2626 using, for instance, a dot script.

2627 The shell will format the output, including the proper use of quoting, so that it is suitable for
 2628 reinput to the shell as commands that achieve the same exporting results.

2629 When no arguments are given, the results are unspecified. |

2630 **EXAMPLES**

2631 Export *PWD* and *HOME* variables:

2632 *export* *PWD* *HOME*

2633 Set and export the *PATH* variable:

2634 *export* *PATH*=/local/bin:\$*PATH*

2635 Save and restore all exported variables:

2636 *export* -p > *temp-file*
 2637 *unset* a lot of variables
 2638 ... processing
 2639 . *temp-file*

2640 **EXIT STATUS**

2641 Zero. |

2642 **2.14.9 readonly — Set Read-only Attribute for Variables**2643 **SYNOPSIS**2644 *readonly* *name*[=*word*]...2645 *readonly* -p2646 **DESCRIPTION**

2647 The variables whose *names* are specified will be given the *readonly* attribute. The values of
 2648 variables with the read-only attribute cannot be changed by subsequent assignment, nor can
 2649 those variables be unset by the *unset* utility.

2650 Some versions of the shell exist that preserve the read-only attribute across separate invocations.
 2651 This specification allows this behaviour, but does not require it.

2652 The *readonly* special built-in supports the XBD specification, **Section 10.2, Utility Syntax**
 2653 **Guidelines**.

2654 When -p is specified, *readonly* will write to the standard output the names and values of all
 2655 read-only variables, in the following format:

2656 "readonly %s=%s\n", <name>, <value>

2657 The shell will format the output, including the proper use of quoting, so that it is suitable for
 2658 reinput to the shell as commands that achieve the same attribute-setting results.

2659 The -p option allows portable access to the values that can be saved and then later restored
 2660 using, for instance, a dot script. (See a related example under *export*.)

2661 **EXAMPLES**

2662 *readonly* HOME PWD

2663 **EXIT STATUS**

2664 Zero.

2665 **2.14.10 return — Return from a Function**2666 **SYNOPSIS**2667 *return* [*n*]2668 **DESCRIPTION**

2669 The *return* utility will cause the shell to stop executing the current function or dot script. If the
2670 shell is not currently executing a function or dot script, the results are unspecified.

2671 The results of returning a number greater than 255 are undefined because of differing practices
2672 in the various historical implementations. Some shells AND out all but the low-order 8 bits;
2673 others allow larger values, but not of unlimited size.

2674 See the discussion of appropriate exit status values under *exit*.

2675 **EXIT STATUS**

2676 The value of the special parameter *?* will be set to *n*, an unsigned decimal integer, or to the exit
2677 status of the last command executed if *n* is not specified. If the value of *n* is greater than 255, the
2678 results are undefined. When *return* is executed in a trap action, the last command is considered
2679 to be the command that executed immediately preceding the trap action.

2680 **2.14.11 set — Set or Unset Options and Positional Parameters**2681 **SYNOPSIS**

```

2682 EX  set [-abCefmnuvx][-h][-o option][argument...]
2683 EX  set [+abCefmnuvx][+h][-o option][argument...]
2684      set --[argument...]
2685 OB  set -[argument...]

```

2686 **DESCRIPTION**

2687 If no options or *arguments* are specified, *set* will write the names and values of all shell variables
 2688 in the collation sequence of the current locale. Each *name* will start on a separate line, using the
 2689 format:

```
2690      "%s=%s\n", <name>, <value>
```

2691 The *value* string will be written with appropriate quoting so that it is suitable for reinput to the
 2692 shell, setting or resetting, as far as possible, the variables that are currently set. Read-only
 2693 variables cannot be reset. See the description of shell quoting in Section 2.2 on page 20.

2694 When options are specified, they will set or unset attributes of the shell, as described below.
 2695 When *arguments* are specified, they will cause positional parameters to be set or unset, as
 2696 described below. Setting or unsetting attributes and positional parameters are not necessarily
 2697 related actions, but they can be combined in a single invocation of *set*.

2698 The *set* special built-in supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**
 2699 except that options can be specified with either a leading hyphen (meaning enable the option) or
 2700 plus sign (meaning disable it).

2701 Implementations support the options in the following list in both their hyphen and plus-sign
 2702 forms. These options can also be specified as options to *sh*.

2703 **-a** When this option is on, the export attribute will be set for each variable to which an
 2704 assignment is performed. (See **Assignment** in the **XBD** specification, **Chapter 2, Glossary**.)
 2705 If the assignment precedes a utility name in a command, the export attribute will not persist
 2706 in the current execution environment after the utility completes, with the exception that
 2707 preceding one of the special built-in utilities will cause the export attribute to persist after
 2708 the built-in has completed. If the assignment does not precede a utility name in the
 2709 command, or if the assignment is a result of the operation of the *getopts* or *read* utilities, the
 2710 export attribute will persist until the variable is unset.

2711 **-b** Cause the shell to notify the user asynchronously of background job completions. The
 2712 following message will be written to standard error:

```
2713      "[%d]%c %s%s\n", <job-number>, <current>, <status>, <job-name>
```

2714 where the fields are as follows:

2715	<current>	The character "+" identifies the job that would be used as a default for the
2716		<i>fg</i> or <i>bg</i> utilities; this job can also be specified using the <i>job_id</i> %+ or %%.
2717		The character "-" identifies the job that would become the default if the
2718		current default job were to exit; this job can also be specified using the
2719		<i>job_id</i> %-. For other jobs, this field is a space character. At most one job
2720		can be identified with "+" and at most one job can be identified with "-".
2721		If there is any suspended job, then the current job will be a suspended job.
2722		If there are at least two suspended jobs, then the previous job will also be
2723		a suspended job.

2724		< <i>job-number</i> >	A number that can be used to identify the process group to the <i>wait</i> , <i>fg</i> , <i>bg</i> and <i>kill</i> utilities. Using these utilities, the job can be identified by prefixing the job number with "%".	
2725				
2726				
2727		< <i>status</i> >	Unspecified.	
2728		< <i>job-name</i> >	Unspecified.	
2729			When the shell notifies the user a job has been completed, it may remove the job's process ID from the list of those known in the current shell execution environment; see Section 2.9.3 on page 50. Asynchronous notification will not be enabled by default.	
2730				
2731				
2732		-C	(Upper-case C.) Prevent existing files from being overwritten by the shell's ">" redirection operator (see Section 2.7.2 on page 41); the > redirection operator will override this <i>noclobber</i> option for an individual file.	
2733				
2734				
2735		-e	When this option is on, if a simple command fails for any of the reasons listed in Section 2.8.1 on page 44 or returns an exit status value >0, and is not part of the compound list following a while , until or if keyword, and is not a part of an AND or OR list, and is not a pipeline preceded by the "!" reserved word, then the shell will immediately exit.	
2736				
2737				
2738				
2739		-f	The shell will disable pathname expansion.	
2740	EX	-h	Locate and remember utilities invoked by functions as those functions are defined (the utilities are normally located when the function is executed).	
2741				
2742		-m	All jobs are run in their own process groups. Immediately before the shell issues a prompt after completion of the background job, a message reporting the exit status of the background job will be written to standard error. If a foreground job stops, the shell will write a message to standard error to that effect, formatted as described by the <i>jobs</i> utility. In addition, if a job changes status other than exiting (for example, if it stops for input or output or is stopped by a SIGSTOP signal), the shell will write a similar message immediately prior to writing the next prompt. This option is enabled by default for interactive shells.	
2743				
2744				
2745				
2746				
2747				
2748				
2749				
2750		-n	The shell will read commands but not execute them; this can be used to check for shell script syntax errors. An interactive shell may ignore this option.	
2751				
2752		-o	<i>option</i>	
2753			Set various options, many of which are equivalent to the single option letters. The following values of <i>option</i> are supported:	
2754				
2755		allexport	Equivalent to -a.	
2756		errexit	Equivalent to -e.	
2757		ignoreeof	Prevent an interactive shell from exiting on end-of-file. This setting prevents accidental logouts when control-D is entered. A user must explicitly <i>exit</i> to leave the interactive shell.	
2758				
2759				
2760		monitor	Equivalent to -m.	
2761		noclobber	Equivalent to -C (upper-case C).	
2762		noglob	Equivalent to -f.	
2763		noexec	Equivalent to -n.	
2764	EX	nolog	Prevent the entry of function definitions into the command history. See Command History List on page 658.	
2765				

2766	notify	Equivalent to -b .
2767	nounset	Equivalent to -u .
2768	verbose	Equivalent to -v .
2769	vi	Allow shell command-line editing using the built-in <i>vi</i> editor. Enabling vi
2770		mode disables any other command-line editing mode provided as an
2771		implementation extension.
2772		It need not be possible to set vi mode on for certain block-mode terminals.
2773	xtrace	Equivalent to -x .
2774	-u	The shell will write a message to standard error when it tries to expand a variable that is not
2775		set and immediately exit. An interactive shell will not exit.
2776	-v	The shell will write its input to standard error as it is read.
2777	-x	The shell will write to standard error a trace for each command after it expands the
2778		command and before it executes it.
2779		The default for all these options is off (unset) unless the shell was invoked with them on (see <i>sh</i>).
2780		All the positional parameters will be unset before any new values are assigned.
2781		The remaining arguments will be assigned in order to the positional parameters. The special
2782		parameter # will be set to reflect the number of positional parameters.
2783		The special argument -- immediately following the <i>set</i> command name can be used to delimit
2784		the arguments if the first argument begins with + or - , or to prevent inadvertent listing of all
2785		shell variables when there are no arguments. The command <i>set --</i> without <i>argument</i> will unset
2786		all positional parameters and set the special parameter # to zero.
2787	OB	In the obsolescent version, the <i>set</i> command name followed by - with no other arguments will
2788		turn off the -v and -x options without changing the positional parameters. The <i>set</i> command
2789		name followed by - with other arguments will turn off the -v and -x options and assign the
2790		arguments to the positional parameters in order.
2791	EXAMPLES	
2792	Write out all variables and their values:	
2793	<code>set</code>	
2794	Set \$1, \$2 and \$3 and set \$# to 3:	
2795	<code>set c a b</code>	
2796	Turn on the -x and -v options:	
2797	<code>set -xv</code>	
2798	Unset all positional parameters:	
2799	<code>set --</code>	
2800	Set \$1 to the value of x , even if x begins with - or + :	
2801	<code>set -- "\$x"</code>	
2802	Set the positional parameters to the expansion of x , even if x expands with a leading - or + :	
2803	<code>set -- \$x</code>	

2804 **EXIT STATUS**

2805 Zero.

|

2806 **2.14.12 shift — Shift Positional Parameters**2807 **SYNOPSIS**2808 *shift* [*n*]2809 **DESCRIPTION**

2810 The positional parameters will be shifted. Positional parameter 1 will be assigned the value of
2811 parameter (1+*n*), parameter 2 will be assigned the value of parameter (2+*n*), and so on. The
2812 parameters represented by the numbers \$# down to \$#-*n*+1 will be unset, and the parameter "#"
2813 will be updated to reflect the new number of positional parameters.

2814 The value *n* will be an unsigned decimal integer less than or equal to the value of the special
2815 parameter "#". If *n* is not given, it will be assumed to be 1. If *n* is 0, the positional and special
2816 parameters will not be changed.

2817 **EXAMPLES**

```
2818 $ set a b c d e  
2819 $ shift 2  
2820 $ echo $*  
2821 c d e
```

2822 **EXIT STATUS**

2823 The exit status will be >0 if *n*>\$#; otherwise, it will be zero.

2824 **2.14.13 times — Write Process Times**2825 **SYNOPSIS**2826 *times*2827 **DESCRIPTION**

2828 Write the accumulated user and system times for the shell and for all of its child processes, in the
 2829 following POSIX locale format:

```
2830      "%dm%fs %dm%fs\n%dm%fs %dm%fs\n", <shell user minutes>,
2831      <shell user seconds>, <shell system minutes>,
2832      <shell system seconds>, <children user minutes>,
2833      <children user seconds>, <children system minutes>
2834      <children system seconds>
```

2835 The four pairs of times correspond to the members of the **XSH** specification `<sys/times.h>` **tms**
 2836 structure as returned by *times()*: *tms_utime*, *tms_stime*, *tms_cutime* and *tms_cstime*, respectively.

2837 **EXAMPLES**

```
2838 $ times
2839 0m0.43s 0m1.11s
2840 8m44.18s 1m43.23s
```

2841 **EXIT STATUS**

2842 Zero.

2843 **2.14.14 trap — Trap Signals**2844 **SYNOPSIS**2845 `trap [action condition ...]`2846 **DESCRIPTION**

2847 If *action* is "-", the shell will reset each *condition* to the default value. If *action* is null (""), the shell
 2848 will ignore each specified *condition* if it arises. Otherwise, the argument *action* will be read and
 2849 executed by the shell when one of the corresponding conditions arises. The action of the trap
 2850 will override a previous action (either default action or one explicitly set). The value of \$? after
 2851 the trap action completes will be the value it had before the trap was invoked.

2852 The condition can be EXIT, 0 (equivalent to EXIT) or a signal specified using a symbolic name,
 2853 without the SIG prefix, as listed in the tables of signal names in the **XSH** specification under
 2854 <signal.h>; for example, HUP, INT, QUIT, TERM. Implementations may permit lower-case
 2855 signal names or names with the SIG prefix as an extension. Setting a trap for SIGKILL or
 2856 SIGSTOP produces undefined results.

2857 The environment in which the shell executes a trap on **EXIT** will be identical to the environment
 2858 immediately after the last command executed before the trap on **EXIT** was taken.

2859 Each time the trap is invoked, the *action* argument will be processed in a manner equivalent to:

2860 `eval "$action"`

2861 Signals that were ignored on entry to a non-interactive shell cannot be trapped or reset, although
 2862 no error need be reported when attempting to do so. An interactive shell may reset or catch
 2863 signals ignored on entry. Traps will remain in place for a given shell until explicitly changed
 2864 with another *trap* command.

2865 When a subshell is entered, traps that are not being ignored are set to the default actions. This
 2866 does not imply that the *trap* command cannot be used within the subshell to set new traps.

2867 The *trap* command with no arguments will write to standard output a list of commands
 2868 associated with each condition. The format is:

2869 `"trap -- %s %s...\n",<action>,<condition> ...`

2870 The shell will format the output, including the proper use of quoting, so that it is suitable for
 2871 reinput to the shell as commands that achieve the same trapping results. For example:

2872 `save_traps=$(trap)`2873 `...`2874 `eval "$save_traps"`

2875 EX OB XSI-conformant systems also allow numeric signal numbers for the conditions, corresponding to
 2876 the following signal names:

2877
2878
2879
2880
2881
2882
2883
2884
2885
2886

	Signal Number	Signal Name	
	1	SIGHUP	
	2	SIGINT	
	3	SIGQUIT	
	6	SIGABRT	
	9	SIGKILL	
	14	SIGALRM	
	15	SIGTERM	

2887

The *trap* special built-in supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

2888

EXAMPLES

2889

Write out a list of all traps and actions:

2890

```
trap
```

2891

Set a trap so the *logout* utility in the *HOME* directory will execute when the shell terminates:

2892

```
trap '$HOME/logout' EXIT
```

2893

or:

2894

```
trap '$HOME/logout' 0
```

2895

Unset traps on INT, QUIT, TERM and EXIT:

2896

```
trap - INT QUIT TERM EXIT
```

2897

EXIT STATUS

2898

If the trap name or number is invalid, a non-zero exit status will be returned; otherwise, zero will be returned. For both interactive and non-interactive shells, invalid signal names or numbers will not be considered a syntax error and will not cause the shell to abort.

2899

2900

2901 **2.14.15 unset — Unset Values and Attributes of Variables and Functions**2902 **SYNOPSIS**2903 `unset [-fv] name . . .`2904 **DESCRIPTION**2905 Each variable or function specified by *name* will be unset.2906 If **-v** is specified, *name* refers to a variable name and the shell will unset it and remove it from the
2907 environment. Read-only variables cannot be unset.2908 If **-f** is specified, *name* refers to a function and the shell will unset the function definition.2909 If neither **-f** nor **-v** is specified, *name* refers to a variable; if a variable by that name does not
2910 exist, it is unspecified whether a function by that name, if any, will be unset.2911 Unsetting a variable or function that was not previously set is not considered an error and will
2912 not cause the shell to abort.2913 The *unset* special built-in supports the **XBD** specification, **Section 10.2, Utility Syntax**
2914 **Guidelines**.

2915 Note that:

2916 `VARIABLE=`2917 is not equivalent to an *unset* of **VARIABLE**; in the example, **VARIABLE** is set to `"`. Also, the
2918 variables that can be *unset* should not be misinterpreted to include the special parameters (see
2919 Section 2.5.2 on page 27).2920 **EXAMPLES**2921 Unset *VISUAL* variable:2922 `unset -v VISUAL`2923 Unset the functions **foo** and **bar**:2924 `unset -f foo bar`2925 **EXIT STATUS**2926 `0` All cases of *name* were successfully unset.2927 `>0` At least one *name* could not be unset.

Utilities

2928

2929 This chapter contains the definitions of the XSI utilities, as follows:

2930 • mandatory utilities that are present on every XSI-conformant system

2931 • software development utilities that are present only on systems supporting the software
2932 development facility; these utilities are marked **DEVELOPMENT**

2933 • the FORTRAN utility that is present only on systems supporting the FORTRAN facility; this
2934 utility is marked **FORTRAN**

2935 • utilities that may be withdrawn and may be present on XSI-conformant systems; these
2936 utilities are marked **LEGACY**.

2937 NAME

2938 admin — create and administer SCCS files (DEVELOPMENT)

2939 SYNOPSIS

```

2940 EX admin -i[name][-n][-a login][-d flag][-f flag][-m mrlist]
2941 [-r rel][-t[name][-y[comment]] newfile
2942 EX admin -n[-a login][-d flag][-f flag][-m mrlist][-t[name]][-y[comment]]
2943 newfile ...
2944 EX admin [-a login][-d flag][-m mrlist][-r rel][-t[name]]
2945 file ...
2946 EX admin -h file ...
2947 EX admin -z file ...

```

2948 DESCRIPTION

2949 The *admin* utility is used to create new SCCS files and change parameters of existing ones. If a
 2950 named file does not exist, it is created, and its parameters are initialised according to the
 2951 specified options. Parameters not initialised by an option are assigned a default value. If a
 2952 named file does exist, parameters corresponding to specified options are changed, and other
 2953 parameters are left as is.

2954 All SCCS filenames must be of the form *s.filename*. New SCCS filenames are given read-only
 2955 permission mode. Write permission in the parent directory is required to create a file. All
 2956 writing done by *admin* is to a temporary *x-file*, named *x.filename* (see *get*) created with read-only
 2957 mode if *admin* is creating a new SCCS file, or created with the same mode as that of the SCCS file
 2958 if the file already exists. After successful execution of *admin*, the SCCS file is removed (if it
 2959 exists), and the *x-file* is renamed with the name of the SCCS file. This ensures that changes are
 2960 made to the SCCS file only if no errors occur.

2961 The *admin* utility also uses a transient lock file (named *z.filename*), which is used to prevent
 2962 simultaneous updates to the SCCS file. See *get* for further information.

2963 OPTIONS

2964 The *admin* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**,
 2965 except that the *-i*, *-t* and *-y* options have optional option-arguments. These optional option-
 2966 arguments cannot be presented as separate arguments. The following options are supported:

2967 **-n** Create a new SCCS file. When *-n* is used without *-i*, the SCCS file is created with
 2968 control information but without any file data.

2969 **-i[name]**

2970 Specify the *name* of a file from which the text for a new SCCS file is to be taken. The
 2971 text constitutes the first delta of the file (see *-r* option for delta numbering scheme). If
 2972 the *-i* option is used, but the *name* option-argument is omitted, the text is obtained by
 2973 reading the standard input. If this option is omitted, the SCCS file is created with
 2974 control information but without any file data. The *-i* option implies the *-n* option.

2975 **-r rel** Specify the *release* into which the initial delta is inserted. If the *-r* option is not used,
 2976 the initial delta is inserted into release 1. The level of the initial delta is always 1 (by
 2977 default initial deltas are named 1.1).

2978 **-t[name]**

2979 Specify the *name* of a file from which descriptive text for the SCCS file is to be taken. In
 2980 the case of existing SCCS files (neither *-i* nor *-n* is specified):

- 2981 • A *-t* option without a *name* option-argument causes the removal of descriptive text
- 2982 (if any) currently in the SCCS file.

2983		• A -t option with a <i>name</i> option-argument causes the text (if any) in the named file to
2984		replace the descriptive text (if any) currently in the SCCS file.
2985	-f flag	Specify a <i>flag</i> , and, possibly, a value for the <i>flag</i> , to be placed in the SCCS file. Several
2986		-f options may be supplied on a single <i>admin</i> command line. The allowable flags and
2987		their values are:
2988	b	Allow use of the -b option on a <i>get</i> command to create branch deltas.
2989	cceil	Specify the highest release (that is, ceiling), a number less than or equal to
2990		9999, which may be retrieved by a <i>get</i> command for editing. The default value
2991		for an unspecified c flag is 9999.
2992	ffloor	Specify the lowest release (that is, floor), a number greater than 0 but less than
2993		9999, which may be retrieved by a <i>get</i> command for editing. The default value
2994		for an unspecified f flag is 1.
2995	dSID	Specify the default delta number (SID) to be used by a <i>get</i> command.
2996	istr	Treat the “No id keywords” message issued by <i>get</i> or <i>delta</i> as a fatal error. In
2997		the absence of this flag, the message is only a warning. The message is issued
2998		if no SCCS identification keywords (see <i>get</i>) are found in the text retrieved or
2999		stored in the SCCS file. If a value is supplied, the keywords must exactly
3000		match the given string; however, the string must contain a keyword, and no
3001		embedded newlines.
3002	j	Allow concurrent <i>get</i> commands for editing on the same SID of an SCCS file.
3003		This allows multiple concurrent updates to the same version of the SCCS file.
3004	llist	Specify a <i>list</i> of releases to which deltas can no longer be made (that is, <i>get -e</i>
3005		against one of these locked releases fails). The <i>list</i> has the following syntax:
3006		<code><list> ::= <range> <list> , <range></code>
3007		<code><range> ::= SID a</code>
3008		The character a in the <i>list</i> is equivalent to specifying all releases for the named
3009		SCCS file.
3010	n	Cause <i>delta</i> to create a null delta in each of those releases (if any) being
3011		skipped when a delta is made in a new release (for example, in making delta
3012		5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas serve as
3013		anchor points so that branch deltas may later be created from them. The
3014		absence of this flag causes skipped releases to be non-existent in the SCCS file,
3015		preventing branch deltas from being created from them in the future.
3016	qtext	Substitute user-definable <i>text</i> for all occurrences of the %Q% keyword in the
3017		SCCS file text retrieved by <i>get</i> .
3018	mmod	Specify the module name of the SCCS file substituted for all occurrences of the
3019		%M% keyword in the SCCS file text retrieved by <i>get</i> . If the m flag is not
3020		specified, the value assigned is the name of the SCCS file with the leading s .
3021		removed.
3022	ttype	Specify the <i>type</i> of module in the SCCS file substituted for all occurrences of
3023		the %Y% keyword in the SCCS file text retrieved by <i>get</i> .

3024 **vpgm** Cause *delta* to prompt for modification request (MR) numbers as the reason
 3025 for creating a delta. The optional value specifies the name of an MR number
 3026 validation program. (If this flag is set when creating an SCCS file, the **m**
 3027 option must also be used even if its value is null.)

3028 **-d flag** Remove (delete) the specified *flag* from an SCCS file. Several **-d** options may be
 3029 supplied on a single *admin* command. See the **-f** option for allowable *flag* names. (The
 3030 **l**list flag gives a *list* of releases to be unlocked. See the **-f** option for further description
 3031 of the **l** flag and the syntax of a *list*.)

3032 **-a login**
 3033 Specify a *login* name, or numerical group ID, to be added to the list of users who may
 3034 make deltas (changes) to the SCCS file. A group ID is equivalent to specifying all *login*
 3035 names common to that group ID. Several **-a** options may be used on a single *admin*
 3036 command line. As many *logins*, or numerical group IDs, as desired may be on the list
 3037 simultaneously. If the list of users is empty, then anyone may add deltas. If *login* or
 3038 group ID is preceded by a **!**, the users so specified are denied permission to make
 3039 deltas.

3040 **-e login**
 3041 Specify a *login* name, or numerical group ID, to be erased from the list of users allowed
 3042 to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to
 3043 specifying all *login* names common to that group ID. Several **-e** options may be used
 3044 on a single *admin* command line.

3045 **-y [comment]**
 3046 Insert the *comment* text into the SCCS file as a comment for the initial delta in a manner
 3047 identical to that of *delta*. In the POSIX locale, omission of the **-y** option results in a
 3048 default comment line being inserted in the form:

3049 "date and time created %s %s by %s", <date>, <time>, <login>

3050 where <date> is expressed in the *date* utility's %y/%m/%d format, <time> in the *date*
 3051 utility's %T format and <login> is the login name of the user creating the file.

3052 **-m mrlist**
 3053 Insert the list of modification request (MR) numbers into the SCCS file as the reason for
 3054 creating the initial delta in a manner identical to *delta*. The **v** flag must be set and the
 3055 MR numbers are validated if the **v** flag has a value (the name of an MR number
 3056 validation program). Diagnostics will occur if the **v** flag is not set or MR validation
 3057 fails.

3058 **-h** Check the structure of the SCCS file and compare the newly computed checksum (the
 3059 sum of all the characters in the SCCS file except those in the first line) with the
 3060 checksum that is stored in the first line of the SCCS file. Appropriate error diagnostics
 3061 are produced.

3062 **-z** Recompute the SCCS file checksum and store it in the first line of the SCCS file (see **-h**
 3063 above). Note that use of this option on a truly corrupted file may prevent future
 3064 detection of the corruption.

3065 **OPERANDS**

3066 The following operands are supported:

3067 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, *admin* behaves
 3068 as though each file in the directory were specified as a named file, except that non-
 3069 SCCS files (last component of the pathname does not begin with s.) and unreadable
 3070 files are silently ignored.

3071 *newfile* A pathname of an SCCS file to be created.

3072 If a single instance *file* or *newfile* is specified as –, the standard input is read; each line of the
 3073 standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files and
 3074 unreadable files are silently ignored.

3075 **STDIN**

3076 The standard input is a text file used only if the **–i** is specified without an option-argument or if
 3077 a *file* or *newfile* operand is specified as –. If the first character of any standard input line is SOH
 3078 (binary 001), the results are unspecified.

3079 **INPUT FILES**

3080 The existing SCCS files are text files of an unspecified format. The file named by the **–i** option's
 3081 *name* option-argument is a text file; if the first character of any line in this file is SOH (binary
 3082 001), the results are unspecified.

3083 **ENVIRONMENT VARIABLES**3084 The following environment variables affect the execution of *admin*:

3085 *LANG* Provide a default value for the internationalisation variables that are unset or null. If
 3086 *LANG* is unset or null, the corresponding value from the implementation-dependent
 3087 default locale will be used. If any of the internationalisation variables contains an
 3088 invalid setting, the utility will behave as if none of the variables had been defined.

3089 *LC_ALL*

3090 If set to a non-empty string value, override the values of all the other
 3091 internationalisation variables.

3092 *LC_CTYPE*

3093 Determine the locale for the interpretation of sequences of bytes of text data as
 3094 characters (for example, single- as opposed to multi-byte characters in arguments and
 3095 input files).

3096 *LC_MESSAGES*

3097 Determine the locale that should be used to affect the format and contents of diagnostic
 3098 messages written to standard error and the contents of the default **–y** comment.

3099 *NLSPATH*

3100 Determine the location of message catalogues for the processing of *LC_MESSAGES*.

3101 **ASYNCHRONOUS EVENTS**

3102 Default.

3103 **STDOUT**

3104 Not used.

3105 **STDERR**

3106 Used only for diagnostic messages.

3107 **OUTPUT FILES**

3108 Any SCCS files created are text files of an unspecified format. During processing of a *file*, a
3109 locking *z-file*, as described in *get*, may be created and deleted.

3110 **EXTENDED DESCRIPTION**

3111 None.

3112 **EXIT STATUS**

3113 The following exit values are returned:

3114 0 Successful completion.

3115 >0 An error occurred.

3116 **CONSEQUENCES OF ERRORS**

3117 Default.

3118 **APPLICATION USAGE**

3119 It is recommended that directories containing SCCS files be writable by the owner only, and that
3120 SCCS files themselves be read-only. The mode of the directories should allow only the owner to
3121 modify SCCS files contained in the directories. The mode of the SCCS files prevents any
3122 modification at all except by SCCS commands.

3123 **EXAMPLES**

3124 None.

3125 **FUTURE DIRECTIONS**

3126 A version of *admin* that fully supports the **XBD** specification, **Section 10.2, Utility Syntax**
3127 **Guidelines** may be introduced in a future issue.

3128 **SEE ALSO**

3129 *delta*, *get*, *prs*, *what*.

3130 **CHANGE HISTORY**

3131 First released in Issue 2.

3132 **Issue 4**

3133 Format reorganised.

3134 Conformance to Utility Syntax Guidelines mandated, with exceptions as noted.

3135 Internationalised environment variable support mandated.

3136 **NAME**

3137 alias — define or display aliases

3138 **SYNOPSIS**3139 alias [*alias-name*[=*string*] ...]3140 **DESCRIPTION**

3141 The *alias* utility creates or redefines alias definitions or writes the values of existing alias
 3142 definitions to standard output. An alias definition provides a string value that replaces a
 3143 command name when it is encountered. See Section 2.3.1 on page 24.

3144 An alias definition affects the current shell execution environment and the execution
 3145 environments of the subshells of the current shell. When used as specified by this specification,
 3146 the alias definition will not affect the parent process of the current shell nor any utility
 3147 environment invoked by the shell. See Section 2.12 on page 63.

3148 **OPTIONS**

3149 None.

3150 **OPERANDS**

3151 The following operands are supported:

3152 *alias-name*

3153 Write the alias definition to standard output.

3154 *alias-name=string*3155 Assign the value of *string* to the alias *alias-name*.

3156 If no operands are given, all alias definitions will be written to standard output.

3157 **STDIN**

3158 Not used.

3159 **INPUT FILES**

3160 None.

3161 **ENVIRONMENT VARIABLES**3162 The following environment variables affect the execution of *alias*:

3163 *LANG* Provide a default value for the internationalisation variables that are unset or null. If
 3164 *LANG* is unset or null, the corresponding value from the implementation-dependent
 3165 default locale will be used. If any of the internationalisation variables contains an
 3166 invalid setting, the utility will behave as if none of the variables had been defined.

3167 *LC_ALL*

3168 If set to a non-empty string value, override the values of all the other
 3169 internationalisation variables.

3170 *LC_CTYPE*

3171 Determine the locale for the interpretation of sequences of bytes of text data as
 3172 characters (for example, single- as opposed to multi-byte characters in arguments).

3173 *LC_MESSAGES*

3174 Determine the locale that should be used to affect the format and contents of diagnostic
 3175 messages written to standard error.

3176 EX *NLSPATH*3177 Determine the location of message catalogues for the processing of *LC_MESSAGES*.

3178 **ASYNCHRONOUS EVENTS**

3179 Default.

3180 **STDOUT**3181 The format for displaying aliases (when no operands or only *name* operands are specified) is:3182 "%s=%s\n", *name*, *value*3183 The *value* string will be written with appropriate quoting so that it is suitable for reinput to the
3184 shell. See the description of shell quoting in Section 2.2 on page 20.3185 **STDERR**

3186 Used only for diagnostic messages.

3187 **OUTPUT FILES**

3188 None.

3189 **EXTENDED DESCRIPTION**

3190 None.

3191 **EXIT STATUS**

3192 The following exit values are returned:

3193 0 Successful completion.

3194 >0 One of the *name* operands specified did not have an alias definition, or an error occurred.3195 **CONSEQUENCES OF ERRORS**

3196 Default.

3197 **APPLICATION USAGE**

3198 None.

3199 **EXAMPLES**3200 1. Change *ls* to give a columnated, more annotated output:

3201 alias ls="ls -CF"

3202 2. Create a simple “redo” command to repeat previous entries in the command history file:

3203 alias r='fc -s'

3204 3. Use 1K units for *du*:

3205 alias du=du\ -k

3206 4. Set up *nohup* so that it can deal with an argument that is itself an alias name:

3207 alias nohup="nohup "

3208 **FUTURE DIRECTIONS**

3209 None.

3210 **SEE ALSO**

3211 Section 2.9.5 on page 54.

3212 **CHANGE HISTORY**

3213 First released in Issue 4.

3214 **NAME**

3215 ar — create and maintain library archives

3216 **SYNOPSIS**3217 EX ar -d[-v][-l] *archive file* ...3218 EX ar -m[-abilv][*posname*] *archive file* ...3219 UN EX ar -p[-v][-s]*archive* [*file* ...]3220 EX ar -q[-clv] *archive file* ...3221 EX ar -r[-cuv][-abil][*posname*]*archive file* ...3222 EX ar -t[-v][-s]*archive* [*file* ...]3223 EX ar -x[-v][-sCT]*archive* [*file* ...]3224 **DESCRIPTION**

3225 The *ar* utility can be used to create and maintain groups of files combined into an archive. Once
 3226 an archive has been created, new files can be added, and existing files can be extracted, deleted
 3227 or replaced. When an archive consists entirely of valid object files, the implementation will
 3228 format the archive so that it is usable as a library for link editing (see *c89*, *cc* and *fort77*). When
 3229 some of the archived files are not valid object files, the suitability of the archive for library use is
 3230 undefined. If an archive file consists entirely of printable files, the entire archive file is printable.

3231 EX When *ar* creates an archive file, it creates administrative information in a format that is portable
 3232 across all machines. When there is at least one object file that *ar* recognises as such in the
 3233 archive, an archive symbol table is created in the archive file and maintained by *ar*; it is used by
 3234 the link editor to search the archive file. Whenever the *ar* utility is used to create or update the
 3235 contents of such an archive, the symbol table is rebuilt. The *-s* option forces the symbol table to
 3236 be rebuilt.

3237 All *file* operands can be pathnames. However, files within archives are named by a filename,
 3238 which is the last component of the pathname used when the file was entered into the archive.
 3239 The comparison of *file* operands to the names of files in archives is performed by comparing the
 3240 last component of the operand to the name of the archive file.

3241 It is unspecified whether multiple files in the archive may be identically named. In the case of
 3242 such files, however, each *file* and *posname* operand will match only the first archive file having a
 3243 name that is the same as the last component of the operand.

3244 **OPTIONS**3245 The *ar* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

3246 The following options are supported:

3247 EX **-a** Position new files in the archive after the file named by the *posname* operand.3248 EX **-b** Position new files in the archive before the file named by the *posname* operand.

3249 **-c** Suppress the diagnostic message that is written to standard error by default when the
 3250 archive file *archive* is created.

3251 EX **-C** Prevent extracted files from replacing like-named files in the file system. This option is
 3252 useful when *-T* is also used, to prevent truncated filenames from replacing files with
 3253 the same prefix.

3254		-d	Delete one or more <i>files</i> from <i>archive</i> .
3255	EX	-i	Position new files in the archive before the file named by the <i>posname</i> operand (equivalent to -b).
3256			
3257	EX	-l	Place temporary files in the local current working directory, rather than in the directory specified by the environment variable <i>TMPDIR</i> or in the default directory. (LEGACY)
3258			
3259	EX	-m	Move the named files. The -a , -b or -i options with the <i>posname</i> operand indicate the position; otherwise, move the files to the end of the archive.
3260			
3261		-p	Write the contents of the <i>files</i> from <i>archive</i> to the standard output. If no <i>files</i> are specified, the contents of all files in the archive will be written in the order of the archive.
3262			
3263			
3264	EX	-q	Quickly append the named files to the end of the archive file. In this case <i>ar</i> does not check whether the added members are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece by piece.
3265			
3266			
3267		-r	Replace or add <i>files</i> to <i>archive</i> . If the archive named by <i>archive</i> does not exist, a new archive file will be created and a diagnostic message will be written to standard error (unless the -c option is specified). If no <i>files</i> are specified and the <i>archive</i> exists, the results are undefined. Files that replace existing files will not change the order of the archive. Files that do not replace existing files will be appended to the archive.
3268			
3269			
3270			
3271			
3272	UN	-s	Force the regeneration of the archive symbol table even if <i>ar</i> is not invoked with an option that will modify the archive file contents. This option is useful to restore the archive symbol table after it has been stripped; see <i>strip</i> .
3273			
3274			
3275		-t	Write a table of contents of <i>archive</i> to the standard output. The files specified by the <i>file</i> operands will be included in the written list. If no <i>file</i> operands are specified, all files in <i>archive</i> will be included in the order of the archive.
3276			
3277			
3278	EX	-T	Allow filename truncation of extracted files whose archive names are longer than the file system can support. By default, extracting a file with a name that is too long is an error; a diagnostic message will be written and the file will not be extracted.
3279			
3280			
3281		-u	Update older files. When used with the -r option, files within the archive will be replaced only if the corresponding <i>file</i> has a modification time that is at least as new as the modification time of the file within the archive.
3282			
3283			
3284		-v	Give verbose output. When used with the option characters -d , -r or -x , write a detailed file-by-file description of the archive creation and maintenance activity, as described in the STDOUT section.
3285			
3286			
3287			When used with -p , write the name of the file to the standard output before writing the file itself to the standard output, as described in the STDOUT section.
3288			
3289			When used with -t , include a long listing of information about the files within the archive, as described in the STDOUT section.
3290			
3291		-x	Extract the files named by the <i>file</i> operands from <i>archive</i> . The contents of the archive file will not be changed. If no <i>file</i> operands are given, all files in the archive will be extracted. If the filename of a file extracted from the archive is longer than that supported in the directory to which it is being extracted, the results are undefined. The modification time of each file extracted will be set to the time the file is extracted from the archive.
3292			
3293			
3294			
3295			
3296			

3297 **OPERANDS**

3298 The following operands are supported:

3299 *archive* A pathname of the archive file.

3300 *file* A pathname. Only the last component will be used when comparing against the names
 3301 of files in the archive. If two or more *file* operands have the same last pathname
 3302 component (basename), the results are unspecified. The implementation's archive
 3303 format will not truncate valid filenames of files added to or replaced in the archive.

3304 EX *posname*

3305 The name of a file in the archive file, used for relative positioning; see options **-m** and
 3306 **-r**.

3307 **STDIN**

3308 Not used.

3309 **INPUT FILES**3310 The input file named by *archive* must be a file in the format created by *ar -r*.3311 **ENVIRONMENT VARIABLES**3312 The following environment variables affect the execution of *ar*:

3313 *LANG* Provide a default value for the internationalisation variables that are unset or null. If
 3314 *LANG* is unset or null, the corresponding value from the implementation-dependent
 3315 default locale will be used. If any of the internationalisation variables contains an
 3316 invalid setting, the utility will behave as if none of the variables had been defined.

3317 *LC_ALL*

3318 If set to a non-empty string value, override the values of all the other
 3319 internationalisation variables.

3320 *LC_CTYPE*

3321 Determine the locale for the interpretation of sequences of bytes of text data as
 3322 characters (for example, single- as opposed to multi-byte characters in arguments and
 3323 input files).

3324 *LC_MESSAGES*

3325 Determine the locale that should be used to affect the format and contents of diagnostic
 3326 messages written to standard error.

3327 *LC_TIME*3328 Determine the format and content for date and time strings written by *ar -tv*.3329 EX *NLSPATH*3330 Determine the location of message catalogues for the processing of *LC_MESSAGES*.3331 *TMPDIR*

3332 Determine the pathname that overrides the default directory for temporary files, if any.

3333 **ASYNCHRONOUS EVENTS**

3334 Default.

3335 **STDOUT**3336 If the **-d** option is used with the **-v** option, the standard output format is:

3337 "d - %s\n", <file>

3338 where <file> is the operand specified on the command line.

3339 If the **-p** option is used with the **-v** option, *ar* will precede the contents of each file with:

3340 `"\n<file>\n\n", <file>`

3341 where *<file>* is the operand specified on the command line, if *file* operands were specified, and

3342 the name of the file in the archive if they were not.

3343 If the **-r** option is used with the **-v** option, and *file* is already in the archive, the standard output

3344 format is:

3345 `"r - %s\n", <file>`

3346 where *<file>* is the operand specified on the command line.

3347 If *file* is being added to the archive with the **-r** option, the standard output format is:

3348 `"a - %s\n", <file>`

3349 where *<file>* is the operand specified on the command line.

3350 If the **-t** option is used, *ar* writes the names of the files to the standard output in the format:

3351 `"%s\n", <file>`

3352 where *<file>* is the operand specified on the command line, if *file* operands were specified, or the

3353 name of the file in the archive if they were not.

3354 If the **-t** option is used with the **-v** option, the standard output format is:

3355 `"%s %u/%u %u %s %d %d:%d %d %s\n", <member mode>, <user ID>,
3356 <group ID>, <number of bytes in member>, <abbreviated month>,
3357 <day-of-month>, <hour>, <minute>, <year>, <file>`

3358 Where:

3359 <i><file></i>	is the operand specified on the command line, if <i>file</i> operands were specified,
3360	or the name of the file in the archive if they were not.
3361 <i><member mode></i>	is formatted the same as the <i><file mode></i> string defined in the STDOUT section
3362	of <i>ls</i> , except that the first character, the <i><entry type></i> , is not used; the string
3363	represents the file mode of the archive member at the time it was added to or
3364	replaced in the archive.

3365 The following represent the last-modification time of a file when it was most recently added to

3366 or replaced in the archive:

3367 <i><abbreviated month></i>	is equivalent to the %b format in <i>date</i> .
3368	
3369 <i><day-of-month></i>	is equivalent to the %e format in <i>date</i> .
3370	
3370 <i><hour></i>	is equivalent to the %H format in <i>date</i> .
3371	
3371 <i><minute></i>	is equivalent to the %M format in <i>date</i> .
3372	
3372 <i><year></i>	is equivalent to the %Y format in <i>date</i> .

3373 When *LC_TIME* does not specify the POSIX locale, a different format and order of presentation

3374 of these fields relative to each other may be used in a format appropriate in the specified locale.

3375 If the **-x** option is used with the **-v** option, the standard output format is:

3376 "x - %s\n", <file>

3377 where <file> is the operand specified on the command line, if *file* operands were specified, or the
3378 name of the file in the archive if they were not.

3379 **STDERR**

3380 Used only for diagnostic messages. The diagnostic message about creating a new archive when
3381 **-c** is not specified will not modify the exit status.

3382 **OUTPUT FILES**

3383 Archives are files with unspecified formats.

3384 **EXTENDED DESCRIPTION**

3385 None.

3386 **EXIT STATUS**

3387 The following exit values are returned:

3388 0 Successful completion.

3389 >0 An error occurred.

3390 **CONSEQUENCES OF ERRORS**

3391 Default.

3392 **APPLICATION USAGE**

3393 None.

3394 **EXAMPLES**

3395 None.

3396 **FUTURE DIRECTIONS**

3397 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this
3398 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the
3399 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when
3400 finalised.

3401 **SEE ALSO**

3402 *c89*, *cc*, *cpio*, *pax*, *strip*. the **XSH** specification <**unistd.h**> description of {POSIX_NO_TRUNC}.

3403 **CHANGE HISTORY**

3404 First released in Issue 2.

3405 **Issue 4**

3406 Aligned with the ISO/IEC 9945-2: 1993 standard. The **-C** and **-T** options are added.

3407 **Issue 5**

3408 FUTURE DIRECTIONS section added.

3409 **NAME**3410 *asa* — interpret carriage-control characters3411 **SYNOPSIS**3412 *asa* [*file* . . .]3413 **DESCRIPTION**3414 The *asa* utility will write its input files to standard output, mapping carriage-control characters from the text files to line-printer control sequences in an implementation-dependent manner.

3416 The first character of every line will be removed from the input, and the following actions will be performed:

3418 If the character removed is:

3419 space The rest of the line will be output without change.

3420 **0** A newline character will be output, then the rest of the input line.3421 **1** One or more implementation-dependent characters that causes an advance to the next page will be output, followed by the rest of the input line.

3423 + The newline character of the previous line will be replaced with one or more implementation-dependent characters that causes printing to return to column position 1, followed by the rest of the input line. If the "+" is the first character in the input, it will have the same effect as the space character.

3427 The action of the *asa* utility is unspecified upon encountering any character other than those listed above as the first character in a line.3429 **OPTIONS**

3430 None.

3431 **OPERANDS**3432 *file* A pathname of a text file used for input. If no *file* operands are specified, the standard input will be used.3434 **STDIN**3435 The standard input will be used only if no *file* operands are specified. See the INPUT FILES section.3437 **INPUT FILES**

3438 The input files must be text files.

3439 **ENVIRONMENT VARIABLES**3440 The following environment variables affect the execution of *asa*:3441 **LANG** Provide a default value for the internationalisation variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.3445 **LC_ALL**

3446 If set to a non-empty string value, override the values of all the other internationalisation variables.

3448 **LC_CTYPE**

3449 Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments and input files).

3452 **LC_MESSAGES**
3453 Determine the locale that should be used to affect the format and contents of diagnostic
3454 messages written to standard error.

3455 EX **NLSPATH**
3456 Determine the location of message catalogues for the processing of *LC_MESSAGES*.

3457 **ASYNCHRONOUS EVENTS**
3458 Default.

3459 **STDOUT**
3460 The standard output will be the text from the input file modified as described in the
3461 DESCRIPTION section.

3462 **STDERR**
3463 None.

3464 **OUTPUT FILES**
3465 None.

3466 **EXTENDED DESCRIPTION**
3467 None.

3468 **EXIT STATUS**
3469 The following exit values are returned:
3470 0 All input files were output successfully.
3471 >0 An error occurred.

3472 **CONSEQUENCES OF ERRORS**
3473 Default.

3474 **APPLICATION USAGE**
3475 None.

3476 **EXAMPLES**
3477 The following command:
3478 `asa file`
3479 permits the viewing of **file** (created by a program using FORTRAN-style carriage control
3480 characters) on a terminal.
3481 The following command:
3482 `a.out | asa | lp`
3483 formats the FORTRAN output of **a.out** and directs it to the printer.

3484 **FUTURE DIRECTIONS**
3485 None.

3486 **SEE ALSO**
3487 *fort77, lp*.

3488 **CHANGE HISTORY**
3489 First released in Issue 4.

3490 NAME

3491 at — execute commands at a later time

3492 SYNOPSIS

3493 at [-m][-f *file*][-q *queuenam*] -t *time*3494 at [-m][-f *file*][-q *queuenam*] *timespec* ...3495 at -r *at_job_id* ...3496 at -l -q *queuenam*3497 at -l [*at_job_id* ...]

3498 DESCRIPTION

3499 The *at* utility reads commands from standard input and groups them together as an *at-job*, to be
3500 executed at a later time.3501 The *at-job* will be executed in a separate invocation of the shell, running in a separate process
3502 group with no controlling terminal, except that the environment variables, current working
3503 directory, file creation mask and other implementation-dependent execution-time attributes in
3504 effect when the *at* utility is executed will be retained and used when the *at-job* is executed.3505 When the *at-job* is submitted, the *at_job_id* and scheduled time are written to standard error.
3506 The *at_job_id* is an identifier that will be a string consisting solely of alphanumeric characters
3507 and the period character. The *at_job_id* is assigned by the system when the job is scheduled such
3508 that it uniquely identifies a particular job.3509 User notification and the processing of the job's standard output and standard error are
3510 described under the *-m* option.3511 EX Users are permitted to use *at* if their name appears in the file */usr/lib/cron/at.allow*. If that file
3512 does not exist, the file */usr/lib/cron/at.deny* is checked to determine if the user should be denied
3513 access to *at*. If neither file exists, only a process with the appropriate privileges is allowed to
3514 submit a job. If only *at.deny* exists and is empty, global usage is permitted. The *at.allow* and
3515 *at.deny* files consist of one user name per line.

3516 OPTIONS

3517 The *at* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

3518 The following options are supported:

3519 *-f file* Specify the pathname of a file to be used as the source of the *at-job*, instead of standard
3520 input.3521 *-l* (The letter ell.) Report all jobs scheduled for the invoking user if no *at_job_id* operands
3522 are specified. If *at_job_ids* are specified, report only information for these jobs. The
3523 output will be written to standard output.3524 *-m* Send mail to the invoking user after the *at-job* has run, announcing its completion.
3525 Standard output and standard error produced by the *at-job* will be mailed to the user as
3526 well, unless redirected elsewhere. Mail will be sent even if the job produces no output.3527 If *-m* is not used, the job's standard output and standard error will be provided to the
3528 EX user by means of mail, unless they are redirected elsewhere; if there is no such output
3529 to provide, the implementation need not notify the user of the job's completion.

3530 **-q** *queuenam*
 3531 Specify in which queue to schedule a job for submission. When used with the **-l**
 3532 option, limit the search to that particular queue. By default, at-jobs will be scheduled in
 3533 queue a. In contrast, queue b is reserved for batch jobs. (See the *batch* utility.) The
 3534 meanings of all other *queuenames* are implementation-dependent.

3535 **-r** Remove the jobs with the specified *at_job_id* operands that were previously scheduled
 3536 by the *at* utility.

3537 **-t** *time* Submit the job to be run at the time specified by the *time* option-argument, which must
 3538 have the format as specified by the *touch* utility.

3539 OPERANDS

3540 The following operands are supported:

3541 **at_job_id**
 3542 The name reported by a previous invocation of the *at* utility at the time the job was
 3543 scheduled.

3544 **timespec** Submit the job to be run at the date and time specified. All of the *timespec* operands are
 3545 interpreted as if they were separated by space characters and concatenated, and are
 3546 parsed as described in the grammar at the end of this section. The date and time are
 3547 interpreted as being in the timezone of the user (as determined by the *TZ* variable),
 3548 unless a timezone name appears as part of *time*, below.

3549 In the POSIX locale, the following describes the three parts of the time specification
 3550 string. All of the values from the LC_TIME categories in the POSIX locale are
 3551 recognised in a case-insensitive manner.

3552 **time** The *time* can be specified as one, two or four digits. One- and two-digit
 3553 numbers are taken to be hours, four-digit numbers to be hours and minutes.
 3554 The time can alternatively be specified as two numbers separated by a colon,
 3555 meaning *hour:minute*. An AM/PM indication (one of the values from the
 3556 **am_pm** keywords in the LC_TIME locale category) can follow the time;
 3557 otherwise, a 24-hour clock time is understood. A timezone name can also
 3558 follow to further qualify the time. The acceptable timezone names are
 3559 implementation-dependent, except that they will be case-insensitive and the
 3560 string **utc** is supported to indicate the time is in Coordinated Universal Time.
 3561 The *time* field can also be one of the following tokens in the POSIX locale:

3562 **midnight** Indicates the time 12:00 am (00:00).

3563 **noon** Indicates the time 12:00 pm.

3564 **now** Indicate the current day and time. Invoking *at <now>* will
 3565 submit an at-job for potentially immediate execution (that is,
 3566 subject only to unspecified scheduling delays).

3567 **date** An optional *date* can be specified as either a month name (one of the values
 3568 from the **mon** or **abmon** keywords in the LC_TIME locale category) followed
 3569 by a day number (and possibly year number preceded by a comma) or a day
 3570 of the week (one of the values from the **day** or **abday** keywords in the
 3571 LC_TIME locale category). Two special days are recognised in the POSIX
 3572 locale:

3573 **today** Indicates the current day.

3574 **tomorrow** Indicates the day following the current day.

3575 If no *date* is given, **today** is assumed if the given time is greater than the
 3576 current time, and **tomorrow** is assumed if it is less. If the given month is less
 3577 than the current month (and no year is given), next year is assumed.

3578 *increment*

3579 The optional *increment* is a number preceded by a plus sign (+) and suffixed by
 3580 one of the following: **minutes**, **hours**, **days**, **weeks**, **months** or **years**. (The
 3581 singular forms will be also accepted.) The keyword **next** is equivalent to an
 3582 increment number of +1. For example, the following are equivalent
 3583 commands:

3584 at 2pm + 1 week
 3585 at 2pm next week

3586 The following grammar describes the precise format of *timespec* in the POSIX locale. The general
 3587 conventions for this style of grammar are described in Section 1.8 on page 10. This formal
 3588 syntax takes precedence over the preceding text syntax description. The longest possible token
 3589 or delimiter will be recognised at a given point. When used in a *timespec*, white space also
 3590 delimits tokens.

3591 %token hr24clock_hr_min
 3592 %token hr24clock_hour
 3593 /*
 3594 A hr24clock_hr_min is a one, two or four digit number. A one or two
 3595 digit number constitutes a hr24clock_hour. A hr24clock_hour may be
 3596 any of the single digits '0' - '9', or may be double digits, ranging
 3597 from "00" - "23". If a hr24clock_hr_min is a four digit number, the
 3598 first two digits must be a valid hr24clock_hour, while the last two
 3599 represent the number of minutes, from "00" - "59".
 3600 */

3601 %token wallclock_hr_min
 3602 %token wallclock_hour
 3603 /*
 3604 A wallclock_hr_min is a one, two or four digit number. A one or two
 3605 digit number constitutes a wallclock_hour. A wallclock_hour may be
 3606 any of the single digits '1' - '9', or may be double digits, ranging
 3607 from "01" - "12". If a wallclock_hr_min is a four digit number, the
 3608 first two digits must be a valid wallclock_hour, while the last two
 3609 represent the number of minutes, from "00" - "59".
 3610 */

3611 %token minute
 3612 /*
 3613 A minute is a one or two digit number whose values can be '0' - '9'
 3614 or "00" - "59".
 3615 */

3616 %token day_number
 3617 /*
 3618 A day_number is a number in the range appropriate for the particular
 3619 month and year specified by month_name and year_number, respectively.
 3620 If no year_number is given, the current year is assumed if the given
 3621 date and time are later this year. If no year_number is given and
 3622 the date and time have already occurred this year and the month is
 3623 not the current month, next year is the assumed year.


```

3624      */
3625      %token year_number
3626      /*
3627          A year_number is a four-digit number representing the year A.D., in
3628          which the at_job is to be run.
3629      */
3630      %token inc_number
3631      /*
3632          The inc_number is the number of times the succeeding increment
3633          period is to be added to the specified date and time.
3634      */
3635      %token timezone_name
3636      /*
3637          The name of an optional timezone suffix to the time field, in an
3638          implementation-dependent format.
3639      */
3640      %token month_name
3641      /*
3642          One of the values from the "mon" or "abmon" keywords in the LC_TIME
3643          locale category.
3644      */
3645      %token day_of_week
3646      /*
3647          One of the values from the "day" or "abday" keywords in the LC_TIME
3648          locale category.
3649      */
3650      %token am_pm
3651      /*
3652          One of the values from the "am_pm" keyword in the LC_TIME locale
3653          category.
3654      */
3655      %start timespec
3656      %%
3657      timespec      : time
3658                     | time date
3659                     | time increment
3660                     | time date increment
3661                     | nowspec
3662                     ;
3663      nowspec       : "now"
3664                     | "now" increment
3665                     ;
3666      time           : hr24clock_hr_min
3667                     | hr24clock_hr_min timezone_name
3668                     | hr24clock_hour ":" minute
3669                     | hr24clock_hour ":" minute timezone_name
3670                     | wallclock_hr_min am_pm
3671                     | wallclock_hr_min am_pm timezone_name

```

```

3672         | wallclock_hour ":" minute am_pm
3673         | wallclock_hour ":" minute am_pm timezone_name
3674         | "noon"
3675         | "midnight"
3676         ;

3677     date      : month_name day_number
3678         | month_name day_number "," year_number
3679         | day_of_week
3680         | "today"
3681         | "tomorrow"
3682         ;

3683     increment  : "+" inc_number inc_period
3684         | "next" inc_period
3685         ;

3686     inc_period : "minute" | "minutes"
3687         | "hour" | "hours"
3688         | "day" | "days"
3689         | "week" | "weeks"
3690         | "month" | "months"
3691         | "year" | "years"
3692         ;

```

3693 STDIN

3694 The standard input must be a text file consisting of commands acceptable to the shell command
 3695 language described in Chapter 2 on page 19. The standard input will only be used if no **-f file**
 3696 option is specified.

3697 INPUT FILES

3698 See the STDIN section.

3699 EX The text files **/usr/lib/cron/at.allow** and **/usr/lib/cron/at.deny** contain user names, one per line, of
 3700 users who are, respectively, authorised or denied access to the *at* and *batch* utilities.

3701 ENVIRONMENT VARIABLES

3702 The following environment variables affect the execution of *at*:

3703 **LANG** Provide a default value for the internationalisation variables that are unset or null. If
 3704 **LANG** is unset or null, the corresponding value from the implementation-dependent
 3705 default locale will be used. If any of the internationalisation variables contains an
 3706 invalid setting, the utility will behave as if none of the variables had been defined.

3707 **LC_ALL**

3708 If set to a non-empty string value, override the values of all the other
 3709 internationalisation variables.

3710 **LC_CTYPE**

3711 Determine the locale for the interpretation of sequences of bytes of text data as
 3712 characters (for example, single- as opposed to multi-byte characters in arguments and
 3713 input files).

3714 **LC_MESSAGES**

3715 Determine the locale that should be used to affect the format and contents of diagnostic
 3716 messages written to standard error and informative messages written to standard
 3717 output.

3718 EX **NLSPATH**
 3719 Determine the location of message catalogues for the processing of *LC_MESSAGES*.

3720 **LC_TIME**
 3721 Determine the format and contents for date and time strings written and accepted by *at*.

3722 **SHELL** Determine a name of a command interpreter to be used to invoke the *at*-job. If the
 3723 variable is unset or null, *sh* will be used. If it is set to a value other than a name for *sh*,
 3724 the implementation will do one of the following: use that shell; use *sh*; use the login
 3725 shell from the user database; or any of the preceding accompanied by a warning
 3726 diagnostic about which was chosen.

3727 **TZ** Determine the timezone. The job will be submitted for execution at the time specified
 3728 by *timespec* or *-t time* relative to the timezone specified by the *TZ* variable. If *timespec*
 3729 specifies a timezone, it will override *TZ*. If *timespec* does not specify a timezone and *TZ*
 3730 is unset or null, an unspecified default timezone will be used.

3731 **ASYNCHRONOUS EVENTS**
 3732 Default.

3733 **STDOUT**
 3734 When standard input is a terminal, prompts of unspecified format for each line of the user input
 3735 described in the STDIN section may be written to standard output.

3736 In the POSIX locale, the following will be written to the standard output for each job when jobs
 3737 are listed in response to the *-l* option:

3738 `"%s\t%s\n", at_job_id, <date>`
 3739 where *<date>* is equivalent in format to the output of:

3740 `date +"%a %b %e %T %Y"`

3741 The date and time written will be adjusted so that they appear in the timezone of the user (as
 3742 determined by the *TZ* variable).

3743 **STDERR**
 3744 The following will be written to standard error when a job has been successfully submitted:

3745 `"job %s at %s\n", at_job_id, <date>`
 3746 where *<date>* has the same format as is described in the STDOUT section. Neither this, nor
 3747 warning messages concerning the selection of the command interpreter, are considered a
 3748 diagnostic that changes the exit status.

3749 Diagnostic messages, if any, are written to standard error.

3750 **OUTPUT FILES**
 3751 None.

3752 **EXTENDED DESCRIPTION**
 3753 None.

3754 **EXIT STATUS**
 3755 The following exit values are returned:

3756 0 The *at* utility successfully submitted, removed or listed a job or jobs.
 3757 >0 An error occurred.

3758 **CONSEQUENCES OF ERRORS**
 3759 The job will not be scheduled, removed or listed.

3760 **APPLICATION USAGE**

3761 The format of the *at* command line shown here is guaranteed only for the POSIX locale. Other
 3762 cultures may be supported with substantially different interfaces, although implementations are
 3763 encouraged to provide comparable levels of functionality.

3764 Since the commands run in a separate shell invocation, running in a separate process group with
 3765 no controlling terminal, open file descriptors, traps and priority inherited from the invoking
 3766 environment are lost.

3767 Some implementations do not allow substitution of different shells using *SHELL*. System V
 3768 systems, for example, have used the login shell value for the user in */etc/passwd*. To select
 3769 reliably another command interpreter, the user must include it as part of the script, such as:

```
3770      $ at 1800
3771      myshell myscript
3772      job ... at ...
3773      $
```

3774 **EXAMPLES**

3775 1. This sequence can be used at a terminal:

```
3776      at -m 0730 tomorrow
3777      sort < file >outfile
3778      EOT
```

3779 2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a
 3780 command procedure (the sequence of output redirection specifications is significant):

```
3781      at now + 1 hour <<!  
3782      diff file1 file2 2>&1 >outfile | mailx mygroup  
3783      !
```

3784 3. To have a job reschedule itself, *at* can be invoked from within the at-job. For example, this
 3785 daily processing script named **my.daily** will run every day (although *crontab* is a more
 3786 appropriate vehicle for such work):

```
3787      # my.daily runs every day
3788      daily processing
3789      at now tomorrow < my.daily
```

3790 4. The spacing of the three portions of the POSIX locale *timespec* is quite flexible as long as
 3791 there are no ambiguities. Examples of various times and operand presentation include:

```
3792      at 0815am Jan 24
3793      at 8 :15amjan24
3794      at now "+ 1day"
3795      at 5 pm FRIday
3796      at '17
3797      utc+
3798      30minutes'
```

3799 **FUTURE DIRECTIONS**

3800 None.

3801 **SEE ALSO**

3802 *batch*, *crontab*.

3803 **CHANGE HISTORY**

3804 First released in Issue 2.

3805 **Issue 4**

3806 Aligned with the ISO/IEC 9945-2: 1993 standard.

3807 **NAME**

3808 awk — pattern scanning and processing language

3809 **SYNOPSIS**3810 awk [-F *ERE*][-v *assignment*] ... *program* [*argument* ...]3811 awk [-F *ERE*] -v *progfile*] ... [-v *assignment*] ...[*argument* ...]3812 **DESCRIPTION**

3813 The *awk* utility executes programs written in the *awk* programming language, which is
 3814 specialised for textual data manipulation. An *awk* program is a sequence of patterns and
 3815 corresponding actions. When input is read that matches a pattern, the action associated with
 3816 that pattern will be carried out.

3817 Input is interpreted as a sequence of records. By default, a record is a line, but this can be
 3818 changed by using the **RS** built-in variable. Each record of input is matched in turn against each
 3819 pattern in the program. For each pattern matched, the associated action will be executed.

3820 The *awk* utility interprets each input record as a sequence of fields where, by default, a field is a
 3821 string of non-blank characters. This default white-space field delimiter can be changed by using
 3822 the **FS** built-in variable or the **-F *ERE***. The *awk* utility denotes the first field in a record \$1, the
 3823 second \$2, and so on. The symbol \$0 refers to the entire record; setting any other field will cause
 3824 the reevaluation of \$0. Assigning to \$0 will reset the values of all other fields and the **NF** built-in
 3825 variable.

3826 **OPTIONS**3827 The *awk* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

3828 The following options are supported:

3829 **-F *ERE***

3830 Define the input field separator to be the extended regular expression *ERE*, before any
 3831 input is read; see **Regular Expressions** on page 119.

3832 **-f *progfile***

3833 Specifies the pathname of the file *progfile* containing an *awk* program. If multiple
 3834 instances of this option are specified, the concatenation of the files specified as *progfile*
 3835 in the order specified will be the *awk* program. The *awk* program can alternatively be
 3836 specified in the command line as a single argument.

3837 **-v *assignment***

3838 The *assignment* argument must be in the same form as an *assignment* operand. The
 3839 specified variable assignment will occur prior to executing the *awk* program, including
 3840 the actions associated with **BEGIN** patterns (if any). Multiple occurrences of this
 3841 option can be specified.

3842 **OPERANDS**

3843 The following operands are supported:

3844 *program*

3845 If no **-f** option is specified, the first operand to *awk* will be the text of the *awk* program.
 3846 The application will supply the *program* operand as a single argument to *awk*. If the
 3847 text does not end in a newline character, *awk* will interpret the text as if it did.

3848 *argument*3849 Either of the following two types of *argument* can be intermixed:

3850 *file* A pathname of a file that contains the input to be read, which is matched
 3851 against the set of patterns in the program. If no *file* operands are specified, or
 3852 if a *file* operand is "-", the standard input will be used.

3853 *assignment*

3854 An operand that begins with an underscore or alphabetic character from the
 3855 portable character set (see the table in the **XBD** specification, **Section 4.1,**
 3856 **Portable Character Set**), followed by a sequence of underscores, digits and
 3857 alphabetic characters from the portable character set, followed by the "=" character will
 3858 specify a variable assignment rather than a pathname. The characters before
 3859 the "=" represent the name of an *awk* variable; if that name is an *awk* reserved
 3860 word (see **Grammar** on page 127) the behaviour is undefined. The characters
 3861 following the equal sign will be interpreted as if they appeared in the *awk*
 3862 program preceded and followed by a double-quote (") character, as a **STRING**
 3863 token (see **Grammar** on page 127), except that if the last character is an
 3864 unescaped backslash, it will be interpreted as a literal backslash rather than as
 3865 the first character of the sequence \". The variable will be assigned the value
 3866 of that **STRING** token. If that value is considered a *numeric string* (see
 3867 **Expressions in awk** on page 113), the variable will also be assigned its
 3868 numeric value. Each such variable assignment will occur just prior to the
 3869 processing of the following *file*, if any. Thus, an assignment before the first *file*
 3870 argument will be executed after the **BEGIN** actions (if any), while an
 3871 assignment after the last *file* argument will occur before the **END** actions (if
 3872 any). If there are no *file* arguments, assignments will be executed before
 3873 processing the standard input.

3874 **STDIN**

3875 The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".
 3876 See the INPUT FILES section.

3877 **INPUT FILES**3878 Input files to the *awk* program from any of the following sources:

- 3879 • any *file* operands or their equivalents, achieved by modifying the *awk* variables **ARGV** and
- 3880 **ARGC**
- 3881 • standard input in the absence of any *file* operands
- 3882 • arguments to the **getline** function

3883 must be text files. Whether the variable **RS** is set to a value other than a newline character or
 3884 not, for these files, implementations support records terminated with the specified separator up
 3885 to {LINE_MAX} bytes and may support longer records.

3886 If **-f progfile** is specified, the files named by each of the *progfile* option-arguments must be text
 3887 files containing an *awk* program.

3888 **ENVIRONMENT VARIABLES**3889 The following environment variables affect the execution of *awk*:

3890 **LANG** Provide a default value for the internationalisation variables that are unset or null. If
 3891 **LANG** is unset or null, the corresponding value from the implementation-dependent
 3892 default locale will be used. If any of the internationalisation variables contains an
 3893 invalid setting, the utility will behave as if none of the variables had been defined.

3894 **LC_ALL**
 3895 If set to a non-empty string value, override the values of all the other
 3896 internationalisation variables.

3897 **LC_COLLATE**
 3898 Determine the locale for the behaviour of ranges, equivalence classes and multi-
 3899 character collating elements within regular expressions and in comparisons of string
 3900 values.

3901 **LC_CTYPE**
 3902 Determine the locale for the interpretation of sequences of bytes of text data as
 3903 characters (for example, single- versus multi-byte characters in arguments and input
 3904 files), the behaviour of character classes within regular expressions, the identification of
 3905 characters as letters, and the mapping of upper- and lower-case characters for the
 3906 **toupper** and **tolower** functions.

3907 **LC_MESSAGES**
 3908 Determine the locale that should be used to affect the format and contents of diagnostic
 3909 messages written to standard error.

3910 **LC_NUMERIC**
 3911 Determine the radix character used when interpreting numeric input, performing
 3912 conversions between numeric and string values and formatting numeric output.
 3913 Regardless of locale, the period character (the decimal-point character of the POSIX
 3914 locale) is the decimal-point character recognised in processing *awk* programs (including
 3915 assignments in command-line arguments).

3916 EX **NLSPATH**
 3917 Determine the location of message catalogues for the processing of **LC_MESSAGES**.

3918 **PATH** Determine the search path when looking for commands executed by *system(expr)*, or
 3919 input and output pipes. See the **XBD** specification, **Chapter 6, Environment Variables**.

3920 In addition, all environment variables will be visible via the *awk* variable **ENVIRON**.

3921 **ASYNCHRONOUS EVENTS**
 3922 Default.

3923 **STDOUT**
 3924 The nature of the output files depends on the *awk* program.

3925 **STDERR**
 3926 Used only for diagnostic messages.

3927 **OUTPUT FILES**
 3928 The nature of the output files depends on the *awk* program.

3929 **EXTENDED DESCRIPTION**

3930 **Overall Program Structure**
 3931 An *awk* program is composed of pairs of the form:
 3932 `pattern { action }`
 3933 Either the pattern or the action (including the enclosing brace characters) can be omitted.
 3934 A missing pattern will match any record of input, and a missing action will be equivalent to an
 3935 action that writes the matched record of input to standard output.

3936 Execution of the *awk* program starts by first executing the actions associated with all **BEGIN**
3937 patterns in the order they occur in the program. Then each *file* operand (or standard input if no
3938 files were specified) will be processed in turn by reading data from the file until a record
3939 separator is seen (a newline character by default), splitting the current record into fields using
3940 the current value of **FS** according to the rules in **Regular Expressions** on page 119, evaluating
3941 each pattern in the program in the order of occurrence, and executing the action associated with
3942 each pattern that matches the current record. The action for a matching pattern will be executed
3943 before evaluating subsequent patterns. Last, the actions associated with all **END** patterns will
3944 be executed in the order they occur in the program.

3945 **Expressions in awk**

3946 Expressions describe computations used in *patterns* and *actions*. In the following table, valid
3947 expression operations are given in groups from highest precedence first to lowest precedence
3948 last, with equal-precedence operators grouped between horizontal lines. In expression
3949 evaluation, where the grammar is formally ambiguous, higher precedence operators will be
3950 evaluated before lower precedence operators. In this table *expr*, *expr1*, *expr2* and *expr3* represent
3951 any expression, while *lvalue* represents any entity that can be assigned to (that is, on the left side
3952 of an assignment operator). The precise syntax of expressions is given in **Grammar** on page 127.

Syntax	Name	Type of Result	Associativity
(<i>expr</i>)	Grouping	type of <i>expr</i>	n/a
<i>\$expr</i>	Field reference	string	n/a
<i>++ lvalue</i>	Pre-increment	numeric	n/a
<i>-- lvalue</i>	Pre-decrement	numeric	n/a
<i>lvalue ++</i>	Post-increment	numeric	n/a
<i>lvalue --</i>	Post-decrement	numeric	n/a
<i>expr ^ expr</i>	Exponentiation	numeric	right
<i>! expr</i>	Logical not	numeric	n/a
<i>+ expr</i>	Unary plus	numeric	n/a
<i>- expr</i>	Unary minus	numeric	n/a
<i>expr * expr</i>	Multiplication	numeric	left
<i>expr / expr</i>	Division	numeric	left
<i>expr % expr</i>	Modulus	numeric	left
<i>expr + expr</i>	Addition	numeric	left
<i>expr - expr</i>	Subtraction	numeric	left
<i>expr expr</i>	String concatenation	string	left
<i>expr < expr</i>	Less than	numeric	none
<i>expr <= expr</i>	Less than or equal to	numeric	none
<i>expr != expr</i>	Not equal to	numeric	none
<i>expr == expr</i>	Equal to	numeric	none
<i>expr > expr</i>	Greater than	numeric	none
<i>expr >= expr</i>	Greater than or equal to	numeric	none
<i>expr ~ expr</i>	ERE match	numeric	none
<i>expr !~ expr</i>	ERE non-match	numeric	none
<i>expr in array</i>	Array membership	numeric	left
(<i>index</i>) in <i>array</i>	Multi-dimension array membership	numeric	left
<i>expr && expr</i>	Logical AND	numeric	left
<i>expr expr</i>	Logical OR	numeric	left
<i>expr1 ? expr2 : expr3</i>	Conditional expression	type of selected <i>expr2</i> or <i>expr3</i>	right
<i>lvalue ^= expr</i>	Exponentiation assignment	numeric	right
<i>lvalue %= expr</i>	Modulus assignment	numeric	right
<i>lvalue *= expr</i>	Multiplication assignment	numeric	right
<i>lvalue /= expr</i>	Division assignment	numeric	right
<i>lvalue += expr</i>	Addition assignment	numeric	right
<i>lvalue -= expr</i>	Subtraction assignment	numeric	right
<i>lvalue = expr</i>	Assignment	type of <i>expr</i>	right

Table 3-1 Expressions in Decreasing Precedence in awk

Each expression has either a string value, a numeric value or both. Except as stated for specific contexts, the value of an expression will be implicitly converted to the type needed for the context in which it is used. A string value will be converted to a numeric value by the equivalent of the following calls to functions defined by the ISO C standard:

```
setlocale(LC_NUMERIC, "");
numeric_value = atof(string_value);
```

A numeric value that is exactly equal to the value of an integer will be converted to a string by the equivalent of a call to the **sprintf** function (see **String Functions** on page 124) with the string `%d` as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. Any other numeric value will be converted to a string by the equivalent of a call to the **sprintf** function with the value of the variable **CONVFMT** as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. The result of the conversion is unspecified if the value of **CONVFMT** is not a floating-point format specification. This specification specifies no explicit conversions between numbers and strings. An application can force an expression to be treated as a number by adding zero to it, or can force it to be treated as a string by concatenating the null string ("") to it.

A string value will be considered to be a *numeric string* in the following case:

1. Any leading and trailing blank characters will be ignored.
2. If the first unignored character is a "+" or "-", it will be ignored.
3. If the remaining unignored characters would be lexically recognised as a **NUMBER** token (as described by the lexical conventions in **Grammar** on page 127), the string will be considered a *numeric string*.

If a "-" character is ignored in the above steps, the numeric value of the *numeric string* will be the negation of the numeric value of the recognised **NUMBER** token. Otherwise the numeric value of the *numeric string* will be the numeric value of the recognised **NUMBER** token. Whether or not a string is a *numeric string* will be relevant only in contexts where that term is used in this section.

When an expression is used in a Boolean context, if it has a numeric value, a value of zero is treated as false and any other value is treated as true. Otherwise, a string value of the null string is treated as false and any other value is treated as true. A Boolean context is one of the following:

- the first subexpression of a conditional expression
- an expression operated on by logical NOT, logical AND or logical OR
- the second expression of a **for** statement
- the expression of an **if** statement
- the expression of the **while** clause in either a **while** or **do...while** statement
- an expression used as a pattern (as in Overall Program Structure).

All arithmetic will follow the semantics of floating-point arithmetic as specified by the ISO C standard.

4032 The value of the expression:

4033 $expr1 \wedge expr2$

4034 will be equivalent to the value returned by the ISO C standard function call:

4035 $\text{pow}(expr1, expr2)$

4036 The expression:

4037 $lvalue \wedge= expr$

4038 will be equivalent to the ISO C standard expression:

4039 $lvalue = \text{pow}(lvalue, expr)$

4040 except that *lvalue* will be evaluated only once. The value of the expression:

4041 $expr1 \% expr2$

4042 will be equivalent to the value returned by the ISO C standard function call:

4043 $\text{fmod}(expr1, expr2)$

4044 The expression:

4045 $lvalue \% = expr$

4046 will be equivalent to the ISO C standard expression:

4047 $lvalue = \text{fmod}(lvalue, expr)$

4048 except that *lvalue* will be evaluated only once.

4049 Variables and fields will be set by the assignment statement:

4050 $lvalue = expression$

4051 and the type of *expression* will determine the resulting variable type. The assignment includes
 4052 the arithmetic assignments ($+=$, $-=$, $*=$, $/=$, $\%=$, $\wedge=$, $++$, $--$) all of which produce a numeric result.
 4053 The left-hand side of an assignment and the target of increment and decrement operators can be
 4054 one of a variable, an array with index or a field selector.

4055 The *awk* language supplies arrays that are used for storing numbers or strings. Arrays need not
 4056 be declared. They are initially empty, and their sizes will change dynamically. The subscripts,
 4057 or element identifiers, are strings, providing a type of associative array capability. An array
 4058 name followed by a subscript within square brackets can be used as an *lvalue* and thus as an
 4059 expression, as described in the grammar (see **Grammar** on page 127). Unsubscripted array
 4060 names can be used in only the following contexts:

- 4061 • a parameter in a function definition or function call
- 4062 • the **NAME** token following any use of the keyword **in** as specified in the grammar (see
- 4063 **Grammar** on page 127); if the name used in this context is not an array name, the behaviour
- 4064 is undefined.

4065 A valid array *index* consists of one or more comma-separated expressions, similar to the way in
 4066 which multi-dimensional arrays are indexed in some programming languages. Because *awk*
 4067 arrays are really one dimensional, such a comma-separated list will be converted to a single
 4068 string by concatenating the string values of the separate expressions, each separated from the
 4069 other by the value of the **SUBSEP** variable. Thus, the following two index operations will be
 4070 equivalent:

```

4071     var[expr1, expr2, ... exprn]
4072     var[expr1 SUBSEP expr2 SUBSEP ... SUBSEP exprn]

```

4073 A multi-dimensioned *index* used with the **in** operator must be parenthesised. The **in** operator,
 4074 which tests for the existence of a particular array element, will not cause that element to exist.
 4075 Any other reference to a non-existent array element will automatically create it.

4076 Comparisons (with the "<", "<=", "!=", "==", ">" and ">=" operators) will be made numerically if
 4077 both operands are numeric or if one is numeric and the other has a string value that is a numeric
 4078 string. Otherwise, operands will be converted to strings as required and a string comparison
 4079 will be made using the locale-specific collation sequence. The value of the comparison
 4080 expression will be 1 if the relation is true, or 0 if the relation is false.

4081 Variables and Special Variables

4082 Variables can be used in an *awk* program by referencing them. With the exception of function
 4083 parameters (see **User-defined Functions** on page 126), they are not explicitly declared.
 4084 Uninitialised scalar variables and array elements have both a numeric value of zero and a string
 4085 value of the empty string.

4086 Field variables are designated by a "\$" followed by a number or numerical expression. The effect
 4087 of the field number *expression* evaluating to anything other than a non-negative integer is
 4088 unspecified; uninitialised variables or string values need not be converted to numeric values in
 4089 this context. New field variables can be created by assigning a value to them. References to
 4090 non-existent fields (that is, fields after \$NF), will produce the null string. However, assigning to
 4091 a non-existent field (for example, \$(NF+2) = 5) will increase the value of NF, create any
 4092 intervening fields with the null string as their values and cause the value of \$0 to be recomputed,
 4093 with the fields being separated by the value of OFS. Each field variable will have a string value
 4094 when created. If the string, with any occurrence of the decimal-point character from the current
 4095 locale changed to a period character, would be considered a *numeric string* (see **Expressions in**
 4096 **awk** on page 113), the field variable will also have the numeric value of the *numeric string*.

4097 Implementations support the following other special variables that are set by *awk*:

4098	ARGC	The number of elements in the ARGV array.
4099	ARGV	An array of command line arguments, excluding options and the <i>program</i>
4100		argument, numbered from zero to ARGC-1.
4101		The arguments in ARGV can be modified or added to; ARGC can be altered.
4102		As each input file ends, <i>awk</i> will treat the next non-null element of ARGV , up
4103		to the current value of ARGC-1, inclusive, as the name of the next input file.
4104		Thus, setting an element of ARGV to null means that it will not be treated as
4105		an input file. The name "-" indicates the standard input. If an argument
4106		matches the format of an <i>assignment</i> operand, this argument will be treated as
4107		an assignment rather than a <i>file</i> argument.
4108	CONVFMT	The printf format for converting numbers to strings (except for output
4109		statements, where OFMT is used); "%.6g" by default.
4110	ENVIRON	The variable ENVIRON is an array representing the value of the environment,
4111		as described in the XSH specification under the <i>exec</i> functions. The indices of
4112		the array are strings consisting of the names of the environment variables, and
4113		the value of each array element is a string consisting of the value of that
4114		variable. If the value of an environment variable is considered a <i>numeric string</i>
4115		(see Expressions in awk on page 113), the array element will also have its
4116		numeric value.

4117		In all cases where the behaviour of <i>awk</i> is affected by environment variables	
4118		(including the environment of any commands that <i>awk</i> executes via the	
4119		system function or via pipeline redirections with the print statement, the	
4120		printf statement, or the getline function), the environment used will be the	
4121		environment at the time <i>awk</i> began executing; it is implementation-dependent	
4122		whether any modification of ENVIRON affects this environment.	
4123	FILENAME	A pathname of the current input file. Inside a BEGIN action the value is	
4124		undefined. Inside an END action the value is the name of the last input file	
4125		processed.	
4126	FNR	The ordinal number of the current record in the current file. Inside a BEGIN	
4127		action the value is zero. Inside an END action the value is the number of the	
4128		last record processed in the last file processed.	
4129	FS	Input field separator regular expression; a space character by default.	
4130	NF	The number of fields in the current record. Inside a BEGIN action, the use of	
4131		NF is undefined unless a getline function without a <i>var</i> argument is executed	
4132		previously. Inside an END action, NF will retain the value it had for the last	
4133		record read, unless a subsequent, redirected, getline function without a <i>var</i>	
4134		argument is performed prior to entering the END action.	
4135	NR	The ordinal number of the current record from the start of input. Inside a	
4136		BEGIN action the value is zero. Inside an END action the value is the number	
4137		of the last record processed.	
4138	OFMT	The printf format for converting numbers to strings in output statements (see	
4139		Output Statements on page 122); "%.6g" by default. The result of the	
4140		conversion is unspecified if the value of OFMT is not a floating-point format	
4141		specification.	
4142	OFS	The print statement output field separation; a space character by default.	
4143	ORS	The print statement output record separator; a newline character by default.	
4144	RLENGTH	The length of the string matched by the match function.	
4145	RS	The first character of the string value of RS is the input record separator; a	
4146		newline character by default. If RS contains more than one character, the	
4147		results are unspecified. If RS is null, then records are separated by sequences	
4148		of one or more blank lines, leading or trailing blank lines do not result in	
4149		empty records at the beginning or end of the input, and a newline character is	
4150		always a field separator, no matter what the value of FS is.	
4151	RSTART	The starting position of the string matched by the match function, numbering	
4152		from 1. This is always equivalent to the return value of the match function.	
4153	SUBSEP	The subscript separator string for multi-dimensional arrays; the default value	
4154		is implementation-dependent.	

Regular Expressions

The *awk* utility makes use of the extended regular expression notation (see the **XBD** specification, **Section 7.4, Extended Regular Expressions**) except that it will allow the use of C-language conventions for escaping special characters within the EREs, as specified in the table in the **XBD** specification, **Chapter 3, File Format Notation** (`\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`) and the following table; these escape sequences will be recognised both inside and outside bracket expressions. Note that records need not be separated by newline characters and string constants can contain newline characters, so even the `\n` sequence is valid in *awk* EREs. Using a slash character within the regular expression requires the escaping shown in the following table:

Escape Sequence	Description	Meaning
<code>\"</code>	Backslash quotation-mark	Quotation-mark character
<code>\/</code>	Backslash slash	Slash character
<code>\ddd</code>	A backslash character followed by the longest sequence of one, two or three octal-digit characters (01234567). If all of the digits are 0, (that is, representation of the NUL character), the behaviour is undefined.	The character whose encoding is represented by the one-, two- or three-digit octal integer. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is implementation-dependent. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <code>\</code> for each byte.
<code>\c</code>	A backslash character followed by any character not described in this table or in the table in the XBD specification, Chapter 3, File Format Notation (<code>\</code> , <code>\a</code> , <code>\b</code> , <code>\f</code> , <code>\n</code> , <code>\r</code> , <code>\t</code> , <code>\v</code>)	Undefined

Table 3-2 Escape Sequences in *awk*

A regular expression can be matched against a specific field or string by using one of the two regular expression matching operators, `~` and `!~`. These operators interpret their right-hand operand as a regular expression and their left-hand operand as a string. If the regular expression matches the string, the `~` expression will evaluate to a value of 1, and the `!~` expression will evaluate to a value of 0. (The regular expression matching operation is as defined by the term matched in the **XBD** specification, **Section 7.1, Regular Expression Definitions**, where a match occurs on any part of the string unless the regular expression is limited with the circumflex or dollar sign special characters.) If the regular expression does not match the string, the `~` expression will evaluate to a value of 0, and the `!~` expression will evaluate to a value of 1. If the right-hand operand is any expression other than the lexical token **ERE**, the string value of the expression will be interpreted as an extended regular expression, including the escape conventions described above. Note that these same escape conventions also will be applied in the determining the value of a string literal (the lexical token **STRING**), and thus will be applied a second time when a string literal is used in this context.

When an **ERE** token appears as an expression in any context other than as the right-hand of the `~` or `!~` operator or as one of the built-in function arguments described below, the value of the resulting expression will be the equivalent of:

```
$0 ~ /ere/
```

The *ere* argument to the **gsub**, **match**, **sub** functions, and the *fs* argument to the **split** function (see **String Functions** on page 124) will be interpreted as extended regular expressions. These can be either **ERE** tokens or arbitrary expressions, and will be interpreted in the same manner as the right-hand side of the `~` or `!~` operator.

An extended regular expression can be used to separate fields by using the `-F ERE` option or by assigning a string containing the expression to the built-in variable **FS**. The default value of the **FS** variable will be a single space character. The following describes **FS** behaviour:

1. If **FS** is a single character:
 - a. If **FS** is the space character, skip leading and trailing blank characters; fields will be delimited by sets of one or more blank characters.
 - b. Otherwise, if **FS** is any other character *c*, fields will be delimited by each single occurrence of *c*.
2. Otherwise, the string value of **FS** will be considered to be an extended regular expression. Each occurrence of a sequence matching the extended regular expression will delimit fields.

Except in the **gsub**, **match**, **split** and **sub** built-in functions, regular expression matching will be based on input records; that is, record separator characters (the first character of the value of the variable **RS**, a newline character by default) cannot be embedded in the expression, and no expression will match the record separator character. If the record separator is not a newline character, newline characters embedded in the expression can be matched. In those four built-in functions, regular expression matching will be based on text strings; that is, any character (including the newline character and the record separator) can be embedded in the pattern and an appropriate pattern will match any character. However, in all *awk* regular expression matching, the use of one or more NUL characters in the pattern, input record or text string produces undefined results.

Patterns

A *pattern* is any valid *expression*, a range specified by two expressions separated by comma, or one of the two special patterns **BEGIN** or **END**.

Special Patterns

The *awk* utility recognises two special patterns, **BEGIN** and **END**. Each **BEGIN** pattern will be matched once and its associated action executed before the first record of input is read (except possibly by use of the **getline** function (see **Input/Output and General Functions** on page 125) in a prior **BEGIN** action) and before command line assignment is done. Each **END** pattern will be matched once and its associated action executed after the last record of input has been read. These two patterns will have associated actions.

BEGIN and **END** will not combine with other patterns. Multiple **BEGIN** and **END** patterns are allowed. The actions associated with the **BEGIN** patterns will be executed in the order specified in the program, as are the **END** actions. An **END** pattern can precede a **BEGIN** pattern in a program.

4243 If an *awk* program consists of only actions with the pattern **BEGIN**, and the **BEGIN** action
 4244 contains no **getline** function, *awk* will exit without reading its input when the last statement in
 4245 the last **BEGIN** action is executed. If an *awk* program consists of only actions with the pattern
 4246 **END** or only actions with the patterns **BEGIN** and **END**, the input will be read before the
 4247 statements in the **END** actions are executed.

4248 Expression Patterns

4249 An expression pattern will be evaluated as if it were an expression in a Boolean context. If the
 4250 result is true, the pattern will be considered to match, and the associated action (if any) will be
 4251 executed. If the result is false, the action will not be executed.

4252 Pattern Ranges

4253 A pattern range consists of two expressions separated by a comma; in this case, the action will
 4254 be performed for all records between a match of the first expression and the following match of
 4255 the second expression, inclusive. At this point, the pattern range can be repeated starting at
 4256 input records subsequent to the end of the matched range.

4257 Actions

4258 An action is a sequence of statements as shown in the grammar in **Grammar** on page 127. Any
 4259 single statement can be replaced by a statement list enclosed in braces. The statements in a
 4260 statement list must be separated by newline characters or semicolons, and will be executed
 4261 sequentially in the order that they appear.

4262 The *expression* acting as the conditional in an **if** statement will be evaluated and if it is non-zero
 4263 or non-null, the following *statement* will be executed; otherwise, if **else** is present, the statement
 4264 following the **else** will be executed.

4265 The **if**, **while**, **do ... while**, **for**, **break** and **continue** statements are based on the ISO C standard,
 4266 except that the Boolean expressions are treated as described in **Expressions in awk** on page 113,
 4267 and except in the case of:

```
4268     for (variable in array)
```

4269 which will iterate, assigning each *index* of *array* to *variable* in an unspecified order. The results of
 4270 adding new elements to *array* within such a **for** loop are undefined. If a **break** or **continue**
 4271 statement occurs outside of a loop, the behaviour is undefined.

4272 The **delete** statement will remove an individual array element. Thus, the following code will
 4273 delete an entire array:

```
4274     for (index in array)
4275         delete array[index]
```

4276 The **next** statement will cause all further processing of the current input record to be abandoned.
 4277 The behaviour is undefined if a **next** statement appears or is invoked in a **BEGIN** or **END** action.

4278 The **exit** statement will invoke all **END** actions in the order in which they occur in the program
 4279 source and then terminate the program without reading further input. An **exit** statement inside
 4280 an **END** action will terminate the program without further execution of **END** actions. If an
 4281 expression is specified in an **exit** statement, its numeric value will be the exit status of *awk*,
 4282 unless subsequent errors are encountered or a subsequent **exit** statement with an expression is
 4283 executed.

Output Statements

Both **print** and **printf** statements write to standard output by default. The output is written to the location specified by *output_redirection* if one is supplied, as follows:

```
> expression
>> expression
| expression
```

In all cases, the *expression* will be evaluated to produce a string that is used as a full pathname to write into (for ">" or ">>") or as a command to be executed (for "|"). Using the first two forms, if the file of that name is not currently open, it will be opened, creating it if necessary and using the first form, truncating the file. The output then will be appended to the file. As long as the file remains open, subsequent calls in which *expression* evaluates to the same string value simply will append output to the file. The file remains open until the **close** function (see **Input/Output and General Functions** on page 125) is called with an expression that evaluates to the same string value.

The third form will write output onto a stream piped to the input of a command. The stream will be created if no stream is currently open with the value of *expression* as its command name. The stream created will be equivalent to one created by a call to the **XSH** specification *popen()* function with the value of *expression* as the *command* argument and a value of **w** as the *mode* argument. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value will write output to the existing stream. The stream will remain open until the **close** function (see **Input/Output and General Functions** on page 125) is called with an expression that evaluates to the same string value. At that time, the stream will be closed as if by a call to the **XSH** specification *pclose()* function.

As described in detail by the grammar in **Grammar** on page 127, these output statements take a comma-separated list of *expressions* referred in the grammar by the non-terminal symbols **expr_list**, **print_expr_list** or **print_expr_list_opt**. This list is referred to here as the *expression list*, and each member is referred to as an *expression argument*.

The **print** statement will write the value of each expression argument onto the indicated output stream separated by the current output field separator (see variable **OFS** above), and terminated by the output record separator (see variable **ORS** above). All expression arguments will be taken as strings, being converted if necessary; this conversion will be as described in **Expressions in awk** on page 113, with the exception that the **printf** format in **OFMT** will be used instead of the value in **CONVFMT**. An empty expression list will stand for the whole input record (\$0).

The **printf** statement will produce output based on a notation similar to the File Format Notation used to describe file formats in this specification (see the **XBD** specification, **Chapter 3, File Format Notation**). Output will be produced as specified with the first expression argument as the string *<format>* and subsequent expression arguments as the strings *<arg1>* to *<argn>*, inclusive, with the following exceptions:

1. The *format* will be an actual character string rather than a graphical representation. Therefore, it cannot contain empty character positions. The space character in the *format* string, in any context other than a *flag* of a conversion specification, will be treated as an ordinary character that is copied to the output.
2. If the character set contains a Δ character and that character appears in the *format* string, it will be treated as an ordinary character that is copied to the output.
3. The *escape sequences* beginning with a backslash character will be treated as sequences of ordinary characters that are copied to the output. Note that these same sequences will be

- 4331 interpreted lexically by *awk* when they appear in literal strings, but they will not be treated
4332 specially by the **printf** statement.
- 4333 4. A *field width* or *precision* can be specified as the `"*"` character instead of a digit string. In this
4334 case the next argument from the expression list will be fetched and its numeric value taken
4335 as the field width or precision.
- 4336 5. The implementation will not precede or follow output from the *d* or *u* conversion
4337 specifications with blank characters not specified by the *format* string.
- 4338 6. The implementation will not precede output from the *o* conversion specification with
4339 leading zeros not specified by the *format* string.
- 4340 7. For the *c* conversion specification: if the argument has a numeric value, the character
4341 whose encoding is that value will be output. If the value is zero or is not the encoding of
4342 any character in the character set, the behaviour is undefined. If the argument does not
4343 have a numeric value, the first character of the string value will be output; if the string
4344 does not contain any characters the behaviour is undefined.
- 4345 8. For each conversion specification that consumes an argument, the next expression
4346 argument will be evaluated. With the exception of the *c* conversion, the value will be
4347 converted (according to the rules specified in **Expressions in awk** on page 113) to the
4348 appropriate type for the conversion specification.
- 4349 9. If there are insufficient expression arguments to satisfy all the conversion specifications in
4350 the *format* string, the behaviour is undefined.
- 4351 10. If any character sequence in the *format* string begins with a `%` character, but does not form
4352 a valid conversion specification, the behaviour is unspecified.

4353 Both **print** and **printf** can output at least {LINE_MAX} bytes.

4354 Functions

4355 The *awk* language has a variety of built-in functions: arithmetic, string, input/output and
4356 general.

4357 Arithmetic Functions

4358 The arithmetic functions, except for **int**, are based on the ISO C standard. The behaviour is
4359 undefined in cases where the ISO C standard specifies that an error be returned or that the
4360 behaviour is undefined. Although the grammar (see **Grammar** on page 127) permits built-in
4361 functions to appear with no arguments or parentheses, unless the argument or parentheses are
4362 indicated as optional in the following list (by displaying them within the `[]` brackets), such use is
4363 undefined.

4364 `atan2(y, x)`

4365 Return arctangent of y/x .

4366 `cos(x)` Return cosine of x , where x is in radians.

4367 `sin(x)` Return sine of x , where x is in radians.

4368 `exp(x)` Return the exponential function of x .

4369 `log(x)` Return the natural logarithm of x .

4370 `sqrt(x)`

4371 Return the square root of x .

4372 `int(x)` Truncate its argument to an integer. It will be truncated toward 0 when $x > 0$.
 4373 `rand()` Return a random number n , such that $0 \leq n < 1$.
 4374 `srand([expr])`
 4375 Set the seed value for **rand** to *expr* or use the time of day if *expr* is omitted. The
 4376 previous seed value will be returned.

4377 **String Functions**

4378 The string functions in the following list shall be supported. Although the grammar (see
 4379 **Grammar** on page 127) permits built-in functions to appear with no arguments or parentheses,
 4380 unless the argument or parentheses are indicated as optional in the following list (by displaying
 4381 them within the [] brackets), such use is undefined.

4382 `gsub(ere, repl[, in])`
 4383 Behave like **sub** (see below), except that it will replace all occurrences of the regular
 4384 expression (like the *ed* utility global substitute) in $\$0$ or in the *in* argument, when
 4385 specified.

4386 `index(s, t)`
 4387 Return the position, in characters, numbering from 1, in string *s* where string *t* first
 4388 occurs, or zero if it does not occur at all.

4389 `length([s])`
 4390 Return the length, in characters, of its argument taken as a string, or of the whole
 4391 EX record, $\$0$, if there is no argument. The use of no argument and no parentheses with
 4392 **length** is obsolescent in the ISO/IEC 9945-2:1993 standard; to be fully portable to
 4393 POSIX systems, the application must use `length($0)` for the length of the whole record.
 4394 However, XSI-conformant systems will continue to support this usage indefinitely.

4395 `match(s, ere)`
 4396 Return the position, in characters, numbering from 1, in string *s* where the extended
 4397 regular expression *ere* occurs, or zero if it does not occur at all. **RSTART** will be set to
 4398 the starting position (which is the same as the returned value), zero if no match is
 4399 found; **RLENGTH** will be set to the length of the matched string, -1 if no match is
 4400 found.

4401 `split(s, a[, fs])`
 4402 Split the string *s* into array elements *a*[1], *a*[2], ..., *a*[*n*], and return *n*. The separation
 4403 will be done with the extended regular expression *fs* or with the field separator **FS** if *fs*
 4404 is not given. Each array element will have a string value when created. If the string
 4405 assigned to any array element, with any occurrence of the decimal-point character from
 4406 the current locale changed to a period character, would be considered a *numeric string*
 4407 (see **Expressions in awk** on page 113), the array element will also have the numeric
 4408 value of the *numeric string*. The effect of a null string as the value of *fs* is unspecified.

4409 `sprintf(fmt, expr, expr, ...)`
 4410 Format the expressions according to the **printf** format given by *fmt* and return the
 4411 resulting string.

4412 `sub(ere, repl[, in])`
 4413 Substitute the string *repl* in place of the first instance of the extended regular expression
 4414 *ERE* in string *in* and return the number of substitutions. An ampersand (&) appearing
 4415 in the string *repl* will be replaced by the string from *in* that matches the regular
 4416 expression. For each occurrence of backslash (\) encountered when scanning the string
 4417 *repl* from beginning to end, the next character is taken literally and loses its special

4418 meaning (for example, `\&` will be interpreted as a literal ampersand character). Except
 4419 for `&` and `\`, it is unspecified what the special meaning of any such character is. If *in* is
 4420 specified and it is not an *lvalue* (see **Expressions in awk** on page 113), the behaviour is
 4421 undefined. If *in* is omitted, *awk* will substitute in the current record (`$0`).

4422 `substr(s, m[, n])`

4423 Return the at most *n*-character substring of *s* that begins at position *m*, numbering from
 4424 1. If *n* is missing, the length of the substring will be limited by the length of the string *s*.

4425 `tolower(s)`

4426 Return a string based on the string *s*. Each character in *s* that is an upper-case letter
 4427 specified to have a **tolower** mapping by the LC_CTYPE category of the current locale
 4428 will be replaced in the returned string by the lower-case letter specified by the
 4429 mapping. Other characters in *s* will be unchanged in the returned string.

4430 `toupper(s)`

4431 Return a string based on the string *s*. Each character in *s* that is a lower-case letter
 4432 specified to have a **toupper** mapping by the LC_CTYPE category of the current locale
 4433 will be replaced in the returned string by the upper-case letter specified by the
 4434 mapping. Other characters in *s* will be unchanged in the returned string.

4435 All of the preceding functions that take *ERE* as a parameter expect a pattern or a string valued
 4436 expression that is a regular expression as defined in **Regular Expressions** on page 119.

4437 **Input/Output and General Functions**

4438 The input/output and general functions are:

4439 `close(expression)`

4440 Close the file or pipe opened by a **print** or **printf** statement or a call to **getline** with the
 4441 same string-valued *expression*. The limit on the number of open *expression* arguments is
 4442 implementation-dependent. If the close was successful, the function will return zero;
 4443 otherwise, it will return non-zero.

4444 `expression | getline [var]`

4445 Read a record of input from a stream piped from the output of a command. The stream
 4446 will be created if no stream is currently open with the value of *expression* as its
 4447 command name. The stream created will be equivalent to one created by a call to the
 4448 *popen()* function with the value of *expression* as the *command* argument and a value of *r*
 4449 as the *mode* argument. As long as the stream remains open, subsequent calls in which
 4450 *expression* evaluates to the same string value will read subsequent records from the file.
 4451 The stream will remain open until the **close** function is called with an expression that
 4452 evaluates to the same string value. At that time, the stream will be closed as if by a call
 4453 to the *pclose()* function. If *var* is missing, `$0` and **NR** will be set; otherwise, *var* will be
 4454 set.

4455 The **getline** operator can form ambiguous constructs when there are unparenthesised
 4456 operators (including concatenate) to the left of the `"|"` (to the beginning of the
 4457 expression containing **getline**). In the context of the `"$"` operator, `"|"` behaves as if it
 4458 had a lower precedence than `"$"`. The result of evaluating other operators is
 4459 unspecified, and portable applications must parenthesise properly all such usages.

4460 `getline`

4461 Set `$0` to the next input record from the current input file. This form of **getline** will set
 4462 the **NR**, **NR** and **FNR** variables.

4463 `getline var`
 4464 Set variable *var* to the next input record from the current input file. This form of
 4465 **getline** will set the **FNR** and **NR** variables.

4466 `getline [var] < expression`
 4467 Read the next record of input from a named file. The *expression* will be evaluated to
 4468 produce a string that is used as a full pathname. If the file of that name is not currently
 4469 open, it will be opened. As long as the stream remains open, subsequent calls in which
 4470 *expression* evaluates to the same string value will read subsequent records from the file.
 4471 The file will remain open until the **close** function is called with an expression that
 4472 evaluates to the same string value. If *var* is missing, **\$0** and **NF** will be set; otherwise,
 4473 *var* will be set.

4474 The **getline** operator can form ambiguous constructs when there are unparenthesised
 4475 binary operators (including concatenate) to the right of the "<" (up to the end of the
 4476 expression containing the **getline**). The result of evaluating such a construct is
 4477 unspecified, and portable applications must parenthesise properly all such usages.

4478 `system(expression)`
 4479 Execute the command given by *expression* in a manner equivalent to the **XSH**
 4480 specification `system()` function and return the exit status of the command.

4481 All forms of **getline** will return 1 for successful input, zero for end-of-file, and -1 for an error.

4482 Where strings are used as the name of a file or pipeline, the strings must be textually identical.
 4483 The terminology "same string value" implies that "equivalent strings", even those that differ
 4484 only by space characters, represent different files.

4485 User-defined Functions

4486 The *awk* language also provides user-defined functions. Such functions can be defined as:

4487 `function name(args, ...) { statements }`

4488 A function can be referred to anywhere in an *awk* program; in particular, its use can precede its
 4489 definition. The scope of a function will be global.

4490 Function arguments can be either scalars or arrays; the behaviour is undefined if an array name
 4491 is passed as an argument that the function uses as a scalar, or if a scalar expression is passed as
 4492 an argument that the function uses as an array. Function arguments will be passed by value if
 4493 scalar and by reference if array name. Argument names will be local to the function; all other
 4494 variable names will be global. The same name will not be used as both an argument name and
 4495 as the name of a function or a special *awk* variable. The same name must not be used both as a
 4496 variable name with global scope and as the name of a function. The same name must not be
 4497 used within the same scope both as a scalar variable and as an array.

4498 The number of parameters in the function definition need not match the number of parameters
 4499 in the function call. Excess formal parameters can be used as local variables. If fewer arguments
 4500 are supplied in a function call than are in the function definition, the extra parameters that are
 4501 used in the function body as scalars will be initialised with a string value of the null string and a
 4502 numeric value of zero, and the extra parameters that are used in the function body as arrays will
 4503 be initialised as empty arrays. If more arguments are supplied in a function call than are in the
 4504 function definition, the behaviour is undefined.

4505 When invoking a function, no white space can be placed between the function name and the
 4506 opening parenthesis. Function calls can be nested and recursive calls can be made upon
 4507 functions. Upon return from any nested or recursive function call, the values of all of the calling
 4508 function's parameters will be unchanged, except for array parameters passed by reference. The

4509 **return** statement can be used to return a value. If a **return** statement appears outside of a
 4510 function definition, the behaviour is undefined.

4511 In the function definition, newline characters are optional before the opening brace and after the
 4512 closing brace. Function definitions can appear anywhere in the program where a *pattern-action*
 4513 pair is allowed.

4514 Grammar

4515 The grammar in this section and the lexical conventions in the following section will together
 4516 describe the syntax for *awk* programs. The general conventions for this style of grammar are
 4517 described in Section 1.8 on page 10. A valid program can be represented as the non-terminal
 4518 symbol *program* in the grammar. This formal syntax takes precedence over the preceding text
 4519 syntax description.

```

4520 %token NAME NUMBER STRING ERE
4521 %token FUNC_NAME /* name followed by '(' without white space */

4522 /* Keywords */
4523 %token Begin End
4524 /* 'BEGIN' 'END' */

4525 %token Break Continue Delete Do Else
4526 /* 'break' 'continue' 'delete' 'do' 'else' */

4527 %token Exit For Function If In
4528 /* 'exit' 'for' 'function' 'if' 'in' */

4529 %token Next Print Printf Return While
4530 /* 'next' 'print' 'printf' 'return' 'while' */

4531 /* Reserved function names */
4532 %token BUILTIN_FUNC_NAME
4533 /* one token for the following:
4534 * atan2 cos sin exp log sqrt int rand srand
4535 * gsub index length match split sprintf sub
4536 * substr tolower toupper close system
4537 */
4538 %token GETLINE
4539 /* Syntactically different from other built-ins */

4540 /* Two-character tokens */
4541 %token ADD_ASSIGN SUB_ASSIGN MUL_ASSIGN DIV_ASSIGN MOD_ASSIGN POW_ASSIGN
4542 /* '+=' '-=' '*=' '/=' '%=' '^=' */

4543 %token OR AND NO_MATCH EQ LE GE NE INCR DECR APPEND
4544 /* '|', '&&', '!~', '==', '<=', '>=', '!=', '++', '--', '>>' */

4545 /* One-character tokens */
4546 %token '{' '}' '(' ')' '[' ']' ',' ';' NEWLINE
4547 %token '+' '-' '*' '%' '^' '!' '>' '<' '|' '?' ':' '~' '$' '='

4548 %start program
4549 %%

4550 program : item_list
4551 | actionless_item_list
4552 ;

```

```

4553     item_list      : newline_opt
4554                     | actionless_item_list item terminator
4555                     | item_list           item terminator
4556                     | item_list           action terminator
4557                     ;

4558     actionless_item_list : item_list           pattern terminator
4559                     | actionless_item_list pattern terminator
4560                     ;

4561     item              : pattern action
4562                     | Function NAME      '(' param_list_opt ')'
4563                       newline_opt action
4564                     | Function FUNC_NAME '(' param_list_opt ')'
4565                       newline_opt action
4566                     ;

4567     param_list_opt    : /* empty */
4568                     | param_list
4569                     ;

4570     param_list        : NAME
4571                     | param_list ',' NAME
4572                     ;

4573     pattern           : Begin
4574                     | End
4575                     | expr
4576                     | expr ',' newline_opt expr
4577                     ;

4578     action            : '{' newline_opt
4579                     | '{' newline_opt terminated_statement_list
4580                     | '{' newline_opt unterminated_statement_list
4581                     ;

4582     terminator        : terminator ';'
4583                     | terminator NEWLINE
4584                     |
4585                       ';'
4586                     |
4587                       NEWLINE
4588                     ;

4587     terminated_statement_list : terminated_statement
4588                     | terminated_statement_list terminated_statement
4589                     ;

4590     unterminated_statement_list : unterminated_statement
4591                     | terminated_statement_list unterminated_statement
4592                     ;

4593     terminated_statement : action newline_opt
4594                     | If '(' expr ')' newline_opt terminated_statement
4595                     | If '(' expr ')' newline_opt terminated_statement
4596                       Else newline_opt terminated_statement
4597                     | While '(' expr ')' newline_opt terminated_statement
4598                     | For '(' simple_statement_opt ';'
4599                       expr_opt ';' simple_statement_opt ')' newline_opt

```



```

4600             terminated_statement
4601         | For '(' NAME In NAME ')' newline_opt
4602             terminated_statement
4603         | ';' newline_opt
4604         | terminatable_statement NEWLINE newline_opt
4605         | terminatable_statement ';'          newline_opt
4606         ;

4607     unterminated_statement : terminatable_statement
4608         | If '(' expr ')' newline_opt unterminated_statement
4609         | If '(' expr ')' newline_opt terminated_statement
4610         | Else newline_opt unterminated_statement
4611         | While '(' expr ')' newline_opt unterminated_statement
4612         | For '(' simple_statement_opt ';'
4613         |   expr_opt ';' simple_statement_opt ')' newline_opt
4614         |   unterminated_statement
4615         | For '(' NAME In NAME ')' newline_opt
4616         |   unterminated_statement
4617         ;

4618     terminatable_statement : simple_statement
4619         | Break
4620         | Continue
4621         | Next
4622         | Exit expr_opt
4623         | Return expr_opt
4624         | Do newline_opt terminated_statement While '(' expr ')'
4625         ;

4626     simple_statement_opt : /* empty */
4627         | simple_statement
4628         ;

4629     simple_statement : Delete NAME '[' expr_list ']'
4630         | expr
4631         | print_statement
4632         ;

4633     print_statement : simple_print_statement
4634         | simple_print_statement output_redirection
4635         ;

4636     simple_print_statement : Print print_expr_list_opt
4637         | Print '(' multiple_expr_list ')'
4638         | Printf print_expr_list
4639         | Printf '(' multiple_expr_list ')'
4640         ;

4641     output_redirection : '>'      expr
4642         | APPEND expr
4643         | '|'      expr
4644         ;

4645     expr_list_opt : /* empty */
4646         | expr_list
4647         ;

```

```

4648     expr_list      : expr
4649                     | multiple_expr_list
4650                     ;

4651     multiple_expr_list : expr ',' newline_opt expr
4652                       | multiple_expr_list ',' newline_opt expr
4653                       ;

4654     expr_opt        : /* empty */
4655                     | expr
4656                     ;

4657     expr            : unary_expr
4658                     | non_unary_expr
4659                     ;

4660     unary_expr      : '+' expr
4661                     | '-' expr
4662                     | unary_expr '^'      expr
4663                     | unary_expr '*'      expr
4664                     | unary_expr '/'      expr
4665                     | unary_expr '%'      expr
4666                     | unary_expr '+'      expr
4667                     | unary_expr '-'      expr
4668                     | unary_expr          non_unary_expr
4669                     | unary_expr '<'      expr
4670                     | unary_expr LE      expr
4671                     | unary_expr NE      expr
4672                     | unary_expr EQ      expr
4673                     | unary_expr '>'      expr
4674                     | unary_expr GE      expr
4675                     | unary_expr '~'      expr
4676                     | unary_expr NO_MATCH expr
4677                     | unary_expr In NAME
4678                     | unary_expr AND newline_opt expr
4679                     | unary_expr OR  newline_opt expr
4680                     | unary_expr '?' expr ':' expr
4681                     | unary_input_function
4682                     ;

4683     non_unary_expr  : '(' expr ')'
4684                     | '!' expr
4685                     | non_unary_expr '^'      expr
4686                     | non_unary_expr '*'      expr
4687                     | non_unary_expr '/'      expr
4688                     | non_unary_expr '%'      expr
4689                     | non_unary_expr '+'      expr
4690                     | non_unary_expr '-'      expr
4691                     | non_unary_expr          non_unary_expr
4692                     | non_unary_expr '<'      expr
4693                     | non_unary_expr LE      expr
4694                     | non_unary_expr NE      expr
4695                     | non_unary_expr EQ      expr
4696                     | non_unary_expr '>'      expr
4697                     | non_unary_expr GE      expr

```

```

4698         | non_unary_expr '~'      expr
4699         | non_unary_expr NO_MATCH expr
4700         | non_unary_expr In NAME
4701         | '(' multiple_expr_list ')' In NAME
4702         | non_unary_expr AND newline_opt expr
4703         | non_unary_expr OR  newline_opt expr
4704         | non_unary_expr '?' expr ':' expr
4705         | NUMBER
4706         | STRING
4707         | lvalue
4708         | ERE
4709         | lvalue INCR
4710         | lvalue DECR
4711         | INCR lvalue
4712         | DECR lvalue
4713         | lvalue POW_ASSIGN expr
4714         | lvalue MOD_ASSIGN expr
4715         | lvalue MUL_ASSIGN expr
4716         | lvalue DIV_ASSIGN expr
4717         | lvalue ADD_ASSIGN expr
4718         | lvalue SUB_ASSIGN expr
4719         | lvalue '=' expr
4720         | FUNC_NAME '(' expr_list_opt ')'
4721         | /* no white space allowed before '(' */
4722         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
4723         | BUILTIN_FUNC_NAME
4724         | non_unary_input_function
4725         ;

4726     print_expr_list_opt : /* empty */
4727         | print_expr_list
4728         ;

4729     print_expr_list : print_expr
4730         | print_expr_list ',' newline_opt print_expr
4731         ;

4732     print_expr : unary_print_expr
4733         | non_unary_print_expr
4734         ;

4735     unary_print_expr : '+' print_expr
4736         | '-' print_expr
4737         | unary_print_expr '^'      print_expr
4738         | unary_print_expr '*'      print_expr
4739         | unary_print_expr '/'      print_expr
4740         | unary_print_expr '%'      print_expr
4741         | unary_print_expr '+'      print_expr
4742         | unary_print_expr '-'      print_expr
4743         | unary_print_expr          non_unary_print_expr
4744         | unary_print_expr '~'      print_expr
4745         | unary_print_expr NO_MATCH print_expr
4746         | unary_print_expr In NAME
4747         | unary_print_expr AND newline_opt print_expr

```

```

4748         unary_print_expr OR newline_opt print_expr
4749         unary_print_expr '?' print_expr ':' print_expr
4750     ;

4751     non_unary_print_expr : '(' expr ')'
4752         | '!' print_expr
4753         | non_unary_print_expr '^' print_expr
4754         | non_unary_print_expr '*' print_expr
4755         | non_unary_print_expr '/' print_expr
4756         | non_unary_print_expr '%' print_expr
4757         | non_unary_print_expr '+' print_expr
4758         | non_unary_print_expr '-' print_expr
4759         | non_unary_print_expr non_unary_print_expr
4760         | non_unary_print_expr '~' print_expr
4761         | non_unary_print_expr NO_MATCH print_expr
4762         | non_unary_print_expr In NAME
4763         | '(' multiple_expr_list ')' In NAME
4764         | non_unary_print_expr AND newline_opt print_expr
4765         | non_unary_print_expr OR newline_opt print_expr
4766         | non_unary_print_expr '?' print_expr ':' print_expr
4767         | NUMBER
4768         | STRING
4769         | lvalue
4770         | ERE
4771         | lvalue INCR
4772         | lvalue DECR
4773         | INCR lvalue
4774         | DECR lvalue
4775         | lvalue POW_ASSIGN print_expr
4776         | lvalue MOD_ASSIGN print_expr
4777         | lvalue MUL_ASSIGN print_expr
4778         | lvalue DIV_ASSIGN print_expr
4779         | lvalue ADD_ASSIGN print_expr
4780         | lvalue SUB_ASSIGN print_expr
4781         | lvalue '=' print_expr
4782         | FUNC_NAME '(' expr_list_opt ')'
4783             /* no white space allowed before '(' */
4784         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
4785         | BUILTIN_FUNC_NAME
4786     ;

4787     lvalue : NAME
4788         | NAME '[' expr_list ']'
4789         | '$' expr
4790     ;

4791     non_unary_input_function : simple_get
4792         | simple_get '<' expr
4793         | non_unary_expr '|' simple_get
4794     ;

4795     unary_input_function : unary_expr '|' simple_get
4796     ;

```

```

4797     simple_get      : GETLINE
4798                       | GETLINE lvalue
4799                       ;
4800     newline_opt      : /* empty */
4801                       | newline_opt NEWLINE
4802                       ;

```

This grammar has several ambiguities that are resolved as follows:

- Operator precedence and associativity are as described in Table 3-1 on page 114.
- In case of ambiguity, an **else** will be associated with the most immediately preceding **if** that would satisfy the grammar.
- In some contexts, a slash (/) that is used to surround an ERE could also be the division operator. This is resolved in such a way that wherever the division operator could appear, a slash is assumed to be the division operator. (There is no unary division operator.)

One convention that might not be obvious from the formal grammar is where newline characters are acceptable. There are several obvious placements such as terminating a statement, and a backslash can be used to escape newline characters between any lexical tokens. In addition, newline characters without backslashes can follow a comma, an open brace, logical AND operator (&&), logical OR operator (| |), the **do** keyword, the **else** keyword, and the closing parenthesis of an **if**, **for** or **while** statement. For example:

```

4816     { print  $1,
4817             $2 }

```

Lexical Conventions

The lexical conventions for *awk* programs, with respect to the preceding grammar, are as follows:

1. Except as noted, *awk* will recognise the longest possible token or delimiter beginning at a given point.
2. A comment consists of any characters beginning with the number sign character and terminated by, but excluding the next occurrence of, a newline character. Comments will have no effect, except to delimit lexical tokens.
3. The character newline will be recognised as the token **NEWLINE**.
4. A backslash character immediately followed by a newline character will have no effect.
5. The token **STRING** represents a string constant. A string constant begins with the character ". Within a string constant, a backslash character will be considered to begin an escape sequence as specified in the table in the **XBD** specification, **Chapter 3, File Format Notation** (\, \a, \b, \f, \n, \r, \t, \v). In addition, the escape sequences in Table 3-2 on page 119 will be recognised. A newline character will not occur within a string constant. A string constant will be terminated by the first unescaped occurrence of the character " after the one that begins the string constant. The value of the string will be the sequence of all unescaped characters and values of escape sequences between, but not including, the two delimiting " characters.
6. The token **ERE** represents an extended regular expression constant. An ERE constant begins with the slash character. Within an ERE constant, a backslash character will be considered to begin an escape sequence as specified in the table in the **XBD** specification, **Chapter 3, File Format Notation**. In addition, the escape sequences in Table 3-2 on page

119 will be recognised. A newline character must not occur within an ERE constant. An ERE constant will be terminated by the first unescaped occurrence of the slash character after the one that begins the string constant. The extended regular expression represented by the ERE constant will be the sequence of all unescaped characters and values of escape sequences between, but not including, the two delimiting slash characters.

7. A blank character has no effect, except to delimit lexical tokens or within **STRING** or **ERE** tokens.
8. The token **NUMBER** represents a numeric constant. Its form and numeric value are equivalent to either of the tokens **floating-constant** or **integer-constant** as specified by the ISO C standard, with the following exceptions:
 - a. An integer constant cannot begin with 0x or include the hexadecimal digits a, b, c, d, e, f, A, B, C, D, E or F.
 - b. The value of an integer constant beginning with 0 will be taken in decimal rather than octal.
 - c. An integer constant cannot include a suffix (u, U, l or L).
 - d. A floating constant cannot include a suffix (f, F, l or L).

If the value is too large or too small to be representable, the behaviour is undefined.

9. A sequence of underscores, digits and alphabetic from the portable character set (see the **XBD** specification, **Section 4.1, Portable Character Set**), beginning with an underscore or alphabetic, will be considered a word.
10. The following words are keywords that will be recognised as individual tokens; the name of the token is the same as the keyword:

BEGIN	delete	for	in	printf
END	do	function	next	return
break	else	getline	print	while
continue	exit	if		

11. The following words are names of built-in functions and will be recognised as the token **BUILTIN_FUNC_NAME**:

atan2	index	match	sprintf	substr
close	int	rand	sqrt	system
cos	length	sin	srand	tolower
exp	log	split	sub	toupper
gsub				

The above-listed keywords and names of built-in functions are considered reserved words.

12. The token **NAME** consists of a word that is not a keyword or a name of a built-in function and is not followed immediately (without any delimiters) by the "(" character.
13. The token **FUNC_NAME** consists of a word that is not a keyword or a name of a built-in function, followed immediately (without any delimiters) by the "(" character. The "(" character will not be included as part of the token.

14. The following two-character sequences will be recognised as the named tokens:

Token Name	Sequence	Token Name	Sequence
ADD_ASSIGN	+=	NO_MATCH	!~
SUB_ASSIGN	-=	EQ	==
MUL_ASSIGN	*=	LE	<=
DIV_ASSIGN	/=	GE	>=
MOD_ASSIGN	%=	NE	!=
POW_ASSIGN	^=	INCR	++
OR		DECR	--
AND	&&	APPEND	>>

15. The following single characters will be recognised as tokens whose names are the character:

<newline> { } () [] , ; + - * % ^ ! > < | ? : ~ \$ =

There is a lexical ambiguity between the token **ERE** and the tokens **"/** and **DIV_ASSIGN**. When an input sequence begins with a slash character in any syntactic context where the token **"/** or **DIV_ASSIGN** could appear as the next token in a valid program, the longer of those two tokens that can be recognised will be recognised. In any other syntactic context where the token **ERE** could appear as the next token in a valid program, the token **ERE** will be recognised.

EXIT STATUS

The following exit values are returned:

0 All input files were processed successfully.
 >0 An error occurred.

The exit status can be altered within the program by using an **exit** expression.

CONSEQUENCES OF ERRORS

If any *file* operand is specified and the named file cannot be accessed, *awk* will write a diagnostic message to standard error and terminate without any further action.

If the program specified by either the *program* operand or a *progfile* operand is not a valid *awk* program (as specified in the EXTENDED DESCRIPTION section), the behaviour is undefined.

APPLICATION USAGE

The **index**, **length**, **match** and **substr** functions should not be confused with similar functions in the ISO C standard; the *awk* versions deal with characters, while the ISO C standard deals with bytes.

Because the concatenation operation is represented by adjacent expressions rather than an explicit operator, it is often necessary to use parentheses to enforce the proper evaluation precedence.

EXAMPLES

The *awk* program specified in the command line is most easily specified within single-quotes (for example, *'program'*) for applications using *sh*, because *awk* programs commonly contain characters that are special to the shell, including double-quotes. In the cases where an *awk* program contains single-quote characters, it is usually easiest to specify most of the program as strings within single-quotes concatenated by the shell with quoted single-quote characters. For example:

```
awk '/\'/ { print "quote:", $0 }
```

prints all lines from the standard input containing a single-quote character, prefixed with quote:.

4925 The following are examples of simple *awk* programs:

- 4926 1. Write to the standard output all input lines for which field 3 is greater than 5:

4927 `$3 > 5`

- 4928 2. Write every tenth line:

4929 `(NR % 10) == 0`

- 4930 3. Write any line with a substring matching the regular expression:

4931 `/(G|D)(2[0-9][[:alpha:]]*)/`

- 4932 4. Print any line with a substring containing a G or D, followed by a sequence of digits and
4933 characters. This example uses character classes **digit** and **alpha** to match language-
4934 independent digit and alphabetic characters respectively:

4935 `/(G|D)([[:digit:]][[:alpha:]]*)/`

- 4936 5. Write any line in which the second field matches the regular expression and the fourth
4937 field does not:

4938 `$2 ~ /xyz/ && $4 !~ /xyz/`

- 4939 6. Write any line in which the second field contains a backslash:

4940 `$2 ~ /\//`

- 4941 7. Write any line in which the second field contains a backslash. Note that backslash escapes
4942 are interpreted twice, once in lexical processing of the string and once in processing the
4943 regular expression:

4944 `$2 ~ "\\\\"`

- 4945 8. Write the second to the last and the last field in each line. Separate the fields by a colon:

4946 `{OFS=":";print $(NF-1), $NF}`

- 4947 9. Write the line number and number of fields in each line. The three strings representing the
4948 line number, the colon and the number of fields are concatenated and that string is written
4949 to standard output:

4950 `{print NR ":" NF}`

- 4951 10. Write lines longer than 72 characters:

4952 `length($0) > 72`

- 4953 11. Write first two fields in opposite order separated by the **OFS**:

4954 `{ print $2, $1 }`

- 4955 12. Same, with input fields separated by comma or space and tab characters, or both:

4956 `BEGIN { FS = ",[\t]*|[\t]+" }`
4957 `{ print $2, $1 }`

- 4958 13. Add up first column, print sum and average:

4959 `{s += $1 }`
4960 `END {print "sum is ", s, " average is", s/NR}`

- 4961 14. Write fields in reverse order, one per line (many lines out for each line in):
- 4962 { for (i = NF; i > 0; --i) print \$i }
- 4963 15. Write all lines between occurrences of the strings **start** and **stop**:
- 4964 /start/, /stop/
- 4965 16. Write all lines whose first field is different from the previous one:
- 4966 \$1 != prev { print; prev = \$1 }
- 4967 17. Simulate *echo*:
- 4968 BEGIN {
- 4969 for (i = 1; i < ARGV; ++i)
- 4970 printf("%s%s", ARGV[i], i==ARGC-1?"\n":" ")
- 4971 }
- 4972 18. Write the path prefixes contained in the *PATH* environment variable, one per line:
- 4973 BEGIN {
- 4974 n = split (ENVIRON["PATH"], path, ":")
- 4975 for (i = 1; i <= n; ++i)
- 4976 print path[i]
- 4977 }
- 4978 19. If there is a file named **input** containing page headers of the form:
- 4979 Page #
- 4980 and a file named **program** that contains:
- 4981 /Page/ { \$2 = n++; }
- 4982 { print }
- 4983 then the command line:
- 4984 awk -f program n=5 input
- 4985 will print the file **input**, filling in page numbers starting at 5.
- 4986 **FUTURE DIRECTIONS**
- 4987 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this
- 4988 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the
- 4989 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when
- 4990 finalised.
- 4991 **SEE ALSO**
- 4992 *grep, lex, sed.*
- 4993 **CHANGE HISTORY**
- 4994 First released in Issue 2.
- 4995 **Issue 4**
- 4996 Aligned with the ISO/IEC 9945-2: 1993 standard.
- 4997 **Issue 4, Version 2**
- 4998 The EXAMPLES section is corrected as follows:
- 4999 • In Example 10, the braces are removed.
- 5000 • In Example 17, the invocation of `printf` is corrected.

5001 **Issue 5**

5002 FUTURE DIRECTIONS section added.

|

5003 **NAME**5004 **basename** — return non-directory portion of a pathname5005 **SYNOPSIS**5006 **basename** *string* [*suffix*]5007 **DESCRIPTION**

5008 The *string* operand will be treated as a pathname, as defined in **Pathname**. The string *string* will
 5009 be converted to the filename corresponding to the last pathname component in *string* and then
 5010 the suffix string *suffix*, if present, will be removed. This will be done by performing actions
 5011 equivalent to the following steps in order:

- 5012 1. If *string* is //, it is implementation-dependent whether steps 2 to 5 are skipped or processed.
- 5013 2. If *string* consists entirely of slash characters, *string* will be set to a single slash character. In
 5014 this case, skip steps 3 to 5.
- 5015 3. If there are any trailing slash characters in *string*, they will be removed.
- 5016 4. If there are any slash characters remaining in *string*, the prefix of *string* up to and including
 5017 the last slash character in *string* will be removed.
- 5018 5. If the *suffix* operand is present, is not identical to the characters remaining in *string*, and is
 5019 identical to a suffix of the characters remaining in *string*, the suffix *suffix* will be removed
 5020 from *string*. Otherwise, *string* will not be modified by this step. It will not be considered
 5021 an error if *suffix* is not found in *string*.

5022 The resulting string will be written to standard output.

5023 **OPTIONS**

5024 None.

5025 **OPERANDS**

5026 The following operands are supported:

5027 *string* A string.5028 *suffix* A string.5029 **STDIN**

5030 Not used.

5031 **INPUT FILES**

5032 None.

5033 **ENVIRONMENT VARIABLES**5034 The following environment variables affect the execution of *basename*:

5035 **LANG** Provide a default value for the internationalisation variables that are unset or null. If
 5036 **LANG** is unset or null, the corresponding value from the implementation-dependent
 5037 default locale will be used. If any of the internationalisation variables contains an
 5038 invalid setting, the utility will behave as if none of the variables had been defined.

5039 **LC_ALL**

5040 If set to a non-empty string value, override the values of all the other
 5041 internationalisation variables.

5042 **LC_CTYPE**

5043 Determine the locale for the interpretation of sequences of bytes of text data as
 5044 characters (for example, single- as opposed to multi-byte characters in arguments).

```

5045      LC_MESSAGES
5046          Determine the locale that should be used to affect the format and contents of diagnostic
5047          messages written to standard error.

5048  EX      NLSPATH
5049          Determine the location of message catalogues for the processing of LC_MESSAGES.

5050  ASYNCHRONOUS EVENTS
5051      Default.

5052  STDOUT
5053      The basename utility will write a line to the standard output in the following format:
5054          "%s\n", <resulting string>

5055  STDERR
5056      Used only for diagnostic messages.

5057  OUTPUT FILES
5058      None.

5059  EXTENDED DESCRIPTION
5060      None.

5061  EXIT STATUS
5062      The following exit values are returned:
5063          0   Successful completion.
5064          >0  An error occurred.

5065  CONSEQUENCES OF ERRORS
5066      Default.

5067  APPLICATION USAGE
5068      The definition of pathname specifies implementation-dependent behaviour for pathnames
5069      starting with two slash characters. Therefore, applications must not arbitrarily add slashes to
5070      the beginning of a pathname unless they can ensure that there are more or less than two or are
5071      prepared to deal with the implementation-dependent consequences.

5072  EXAMPLES
5073      If the string string is a valid pathname:
5074          $(basename "string")
5075      produces a filename that could be used to open the file named by string in the directory returned
5076      by:
5077          $(dirname "string")
5078      If the string string is not a valid pathname, the same algorithm is used, but the result need not be
5079      a valid filename. The basename utility is not expected to make any judgements about the validity
5080      of string as a pathname; it just follows the specified algorithm to produce a result string.
5081      The following shell script compiles /usr/src/cmd/cat.c and moves the output to a file named cat
5082      in the current directory when invoked with the argument /usr/src/cmd/cat or with the
5083      argument /usr/src/cmd/cat.c:
5084          c89 $(dirname "$1")/$(basename "$1" .c).c
5085          mv a.out $(basename "$1" .c)

```

5086 **FUTURE DIRECTIONS**

5087 None.

5088 **SEE ALSO**5089 *dirname*, Section 2.5 on page 27.5090 **CHANGE HISTORY**

5091 First released in Issue 2.

5092 **Issue 4**

5093 Aligned with the ISO/IEC 9945-2: 1993 standard.

5094 NAME

5095 batch — execute commands when the system load permits

5096 SYNOPSIS

5097 *batch*

5098 DESCRIPTION

5099 The *batch* utility reads commands to be executed at a later time. It is the equivalent of the
5100 command:

5101 at -q b -m now

5102 where queue b is a special *at* queue, specifically for batch jobs. Batch jobs will be submitted to
5103 the batch queue with no time constraints and run by the system using algorithms, based on
5104 unspecified factors, that may vary with each invocation of *batch*.

5105 EX Users are permitted to use *batch* if their name appears in the file **/usr/lib/cron/at.allow**. If that file
5106 does not exist, the file **/usr/lib/cron/at.deny** is checked to determine if the user should be denied
5107 access to *batch*. If neither file exists, only a process with the appropriate privileges is allowed to
5108 submit a job. If only **at.deny** exists and is empty, global usage is permitted. The **at.allow** and
5109 **at.deny** files consist of one user name per line.

5110 OPTIONS

5111 None.

5112 OPERANDS

5113 None.

5114 STDIN

5115 The standard input must be a text file consisting of commands acceptable to the shell command
5116 language described in Chapter 2 on page 19. The standard input will only be used if no **-f file**
5117 option is specified.

5118 INPUT FILES

5119 EX The text files **/usr/lib/cron/at.allow** and **/usr/lib/cron/at.deny** contain user names, one per line, of
5120 users who are, respectively, authorised or denied access to the *at* and *batch* utilities.

5121 ENVIRONMENT VARIABLES

5122 The following environment variables affect the execution of *batch*:

5123 **LANG** Provide a default value for the internationalisation variables that are unset or null. If
5124 **LANG** is unset or null, the corresponding value from the implementation-dependent
5125 default locale will be used. If any of the internationalisation variables contains an
5126 invalid setting, the utility will behave as if none of the variables had been defined.

5127 **LC_ALL**

5128 If set to a non-empty string value, override the values of all the other
5129 internationalisation variables.

5130 **LC_CTYPE**

5131 Determine the locale for the interpretation of sequences of bytes of text data as
5132 characters (for example, single- as opposed to multi-byte characters in arguments and
5133 input files).

5134 **LC_MESSAGES**

5135 Determine the locale that should be used to affect the format and contents of diagnostic
5136 messages written to standard error and informative messages written to standard
5137 output.

5138 **LC_TIME**
 5139 Determine the format and contents for date and time strings written by *batch*.

5140 EX **NLSPATH**
 5141 Determine the location of message catalogues for the processing of *LC_MESSAGES*.

5142 **SHELL** Determine the name of a command interpreter to be used to invoke the at-job. If the
 5143 variable is unset or null, *sh* will be used. If it is set to a value other than a name for *sh*,
 5144 the implementation will do one of the following: use that shell; use *sh*; use the login
 5145 shell from the user database; any of the preceding accompanied by a warning
 5146 diagnostic about which was chosen.

5147 **TZ** Determine the timezone. The job will be submitted for execution at the time specified
 5148 by *timespec* or *-t time* relative to the timezone specified by the *TZ* variable. If *timespec*
 5149 specifies a timezone, it will override *TZ*. If *timespec* does not specify a timezone and *TZ*
 5150 is unset or null, an unspecified default timezone will be used.

5151 **ASYNCHRONOUS EVENTS**
 5152 Default.

5153 **STDOUT**
 5154 When standard input is a terminal, prompts of unspecified format for each line of the user input
 5155 described in the STDIN section may be written to standard output.

5156 In the POSIX locale, the following will be written to the standard output for each job when jobs
 5157 are listed in response to the *-l* option:

5158 `"%s\t%s\n", at_job_id, <date>`
 5159 where *<date>* is equivalent in format to the output of:

5160 `date +"%a %b %e %T %Y"`

5161 The date and time written will be adjusted so that they appear in the timezone of the user (as
 5162 determined by the *TZ* variable).

5163 **STDERR**
 5164 The following will be written to standard error when a job has been successfully submitted:

5165 `"job %s at %s\n", at_job_id, <date>`
 5166 where *<date>* will have the same format as is described in the STDOUT section. Neither this,
 5167 nor warning messages concerning the selection of the command interpreter, are considered a
 5168 diagnostic that changes the exit status.

5169 Diagnostic messages, if any, are written to standard error.

5170 **OUTPUT FILES**
 5171 None.

5172 **EXTENDED DESCRIPTION**
 5173 None.

5174 **EXIT STATUS**
 5175 The following exit values are returned:

5176 0 Successful completion.
 5177 >0 An error occurred.

5178 **CONSEQUENCES OF ERRORS**
 5179 The job will not be scheduled.

5180 **APPLICATION USAGE**

5181 It may be useful to redirect standard output within the specified commands.

5182 **EXAMPLES**

5183 1. This sequence can be used at a terminal:

```
5184     batch
5185     sort < file >outfile
5186     EOT
```

5187 2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a
5188 command procedure (the sequence of output redirection specifications is significant):

```
5189     batch <<!  
5190     diff file1 file2 2>&l >outfile | mailx mygroup  
5191     !
```

5192 **FUTURE DIRECTIONS**

5193 None.

5194 **SEE ALSO**5195 *at*.5196 **CHANGE HISTORY**

5197 First released in Issue 2.

5198 **Issue 4**

5199 Format reorganised and separated from the *at* description. Aligned with the ISO/IEC 9945-
5200 2: 1993 standard.

5201 **NAME**

5202 bc — arbitrary-precision arithmetic language

5203 **SYNOPSIS**5204 bc [-l] [*file* ...]5205 **DESCRIPTION**

5206 The *bc* utility implements an arbitrary precision calculator. It takes input from any files given,
 5207 then reads from the standard input. If the standard input and standard output to *bc* are attached
 5208 to a terminal, the invocation of *bc* is considered to be *interactive*, causing behavioural constraints
 5209 described in the following sections.

5210 **OPTIONS**5211 The *bc* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

5212 The following option is supported:

5213 -l (The letter ell.) Define the math functions and initialise **scale** to 20, instead of the
 5214 default zero. See the EXTENDED DESCRIPTION section.

5215 **OPERANDS**

5216 The following operands are supported:

5217 *file* A pathname of a text file containing *bc* program statements. After all cases of *file* have
 5218 been read, *bc* will read the standard input.

5219 **STDIN**

5220 See the INPUT FILES section.

5221 **INPUT FILES**

5222 Input files must be text files containing a sequence of comments, statements and function
 5223 definitions that will be executed as they are read.

5224 **ENVIRONMENT VARIABLES**5225 The following environment variables affect the execution of *bc*:

5226 *LANG* Provide a default value for the internationalisation variables that are unset or null. If
 5227 *LANG* is unset or null, the corresponding value from the implementation-dependent
 5228 default locale will be used. If any of the internationalisation variables contains an
 5229 invalid setting, the utility will behave as if none of the variables had been defined.

5230 *LC_ALL*

5231 If set to a non-empty string value, override the values of all the other
 5232 internationalisation variables.

5233 *LC_CTYPE*

5234 Determine the locale for the interpretation of sequences of bytes of text data as
 5235 characters (for example, single- as opposed to multi-byte characters in arguments and
 5236 input files).

5237 *LC_MESSAGES*

5238 Determine the locale that should be used to affect the format and contents of diagnostic
 5239 messages written to standard error.

5240 EX *NLSPATH*5241 Determine the location of message catalogues for the processing of *LC_MESSAGES*.5242 **ASYNCHRONOUS EVENTS**

5243 Default.

5244 **STDOUT**

The output of the *bc* utility is controlled by the program read, and consists of zero or more lines containing the value of all executed expressions without assignments. The radix and precision of the output are controlled by the values of the **obase** and **scale** variables. See the EXTENDED DESCRIPTION section.

5249 **STDERR**

5250 Used only for diagnostic messages.

5251 OUTPUT FILES

5252 None.

5253 EXTENDED DESCRIPTION

5254 **Grammar**

The grammar in this section and the lexical conventions in the following section together describe the syntax for *bc* programs. The general conventions for this style of grammar are described in Section 1.8 on page 10. A valid program can be represented as the non-terminal symbol **program** in the grammar. This formal syntax takes precedence over the preceding text syntax description.

```
5260      %token      EOF NEWLINE STRING LETTER NUMBER
```

```
5261      %token      MUL_OP
5262      / *          ' * ' , ' / ' , ' % '          * /
```

```
5263 %token ASSIGN_OP
5264 /* ' = ' , ' + = ' , ' - = ' , ' * = ' , ' / = ' , ' % = ' , ' ^ = ' */
```

```
5265      %token      REL_OP
5266      /*          '==', '<=', '>=', '!=', '<', '>'          */
```

```
5267      %token      INCR_DECR
5268      / *          '++', '--'          * /
```

```
5269 %token      Define      Break      Quit      Length
5270 /*          'define',    'break',   'quit',    'length'   */
```

```
5271      %token      Return      For      If      While      Sqrt
5272      /*          'return',    'for',    'if',    'while',    'sqrt'    */
```

```
5273      %token      Scale      Ibase      Obase      Auto
5274      /*          'scale',    'ibase',    'obase',    'auto'      */
```

```
5275      %start  program
```

5276 % %

```

5277      program      : EOF
5278      | input_item program
5279      :

```

```
5280         input_item      : semicolon_list NEWLINE
5281         | function
```

```

5282                                     ;
5283     semicolon_list                   : /* empty */
5284                                     | statement
5285                                     | semicolon_list ';' statement
5286                                     | semicolon_list ';'
5287                                     ;
5288     statement_list                   : /* empty */
5289                                     | statement
5290                                     | statement_list NEWLINE
5291                                     | statement_list NEWLINE statement
5292                                     | statement_list ';'
5293                                     | statement_list ';' statement
5294                                     ;
5295     statement                         : expression
5296                                     | STRING
5297                                     | Break
5298                                     | Quit
5299                                     | Return
5300                                     | Return '(' return_expression ')'
5301                                     | For '(' expression ';'
5302                                     |     relational_expression ';'
5303                                     |     expression ')' statement
5304                                     | If '(' relational_expression ')' statement
5305                                     | While '(' relational_expression ')' statement
5306                                     | '{' statement_list '}'
5307                                     ;
5308     function                         : Define LETTER '(' opt_parameter_list ')'
5309                                     | '{' NEWLINE opt_auto_define_list
5310                                     | statement_list '}'
5311                                     ;
5312     opt_parameter_list               : /* empty */
5313                                     | parameter_list
5314                                     ;
5315     parameter_list                   : LETTER
5316                                     | define_list ',' LETTER
5317                                     ;
5318     opt_auto_define_list             : /* empty */
5319                                     | Auto define_list NEWLINE
5320                                     | Auto define_list ';'
5321                                     ;
5322     define_list                      : LETTER
5323                                     | LETTER '[' ']'
5324                                     | define_list ',' LETTER
5325                                     | define_list ',' LETTER '[' ']'

```

```

5326                                     ;
5327     opt_argument_list      : /* empty */
5328                             | argument_list
5329                             ;
5330     argument_list           : expression
5331                             | argument_list ',' expression
5332                             ;
5333     relational_expression   : expression
5334                             | expression REL_OP expression
5335                             ;
5336     return_expression       : /* empty */
5337                             | expression
5338                             ;
5339     expression              : named_expression
5340                             | NUMBER
5341                             | '(' expression ')'
5342                             | LETTER '(' opt_argument_list ')'
5343                             | '-' expression
5344                             | expression '+' expression
5345                             | expression '-' expression
5346                             | expression MUL_OP expression
5347                             | expression '^' expression
5348                             | INCR_DECR named_expression
5349                             | named_expression INCR_DECR
5350                             | named_expression ASSIGN_OP expression
5351                             | Length '(' expression ')'
5352                             | Sqrt '(' expression ')'
5353                             | Scale '(' expression ')'
5354                             ;
5355     named_expression        : LETTER
5356                             | LETTER '[' expression ']'
5357                             | Scale
5358                             | Ibase
5359                             | Obase
5360                             ;

```

5361 Lexical Conventions in bc

5362 The lexical conventions for *bc* programs, with respect to the preceding grammar, are as follows:

- 5363 1. Except as noted, *bc* recognises the longest possible token or delimiter beginning at a given
5364 point.
- 5365 2. A comment consists of any characters beginning with the two adjacent characters */** and
5366 terminated by the next occurrence of the two adjacent characters **/*. Comments have no
5367 effect except to delimit lexical tokens.
- 5368 3. The character newline is recognised as the token **NEWLINE**.

- 5369 4. The token **STRING** represents a string constant; it consists of any characters beginning
 5370 with the double-quote character (") and terminated by another occurrence of the double-
 5371 quote character. The value of the string is the sequence of all characters between, but not
 5372 including, the two double-quote characters. All characters are taken literally from the
 5373 input, and there is no way to specify a string containing a double-quote character. The
 5374 length of the value of each string is limited to {BC_STRING_MAX} bytes.
- 5375 5. A blank character has no effect except as an ordinary character if it appears within a
 5376 **STRING** token, or to delimit a lexical token other than **STRING**.
- 5377 6. The combination of a backslash character immediately followed by a newline character has
 5378 no effect other than to delimit lexical tokens with the following exceptions:
- 5379 • It is interpreted as the character sequence \newline in **STRING** tokens.
 - 5380 • It is ignored as part of a multi-line **NUMBER** token.
- 5381 7. The token **NUMBER** represents a numeric constant. It is recognised by the following
 5382 grammar:
- ```

5383 NUMBER : integer
5384 | '.' integer
5385 | integer '.'
5386 | integer '.' integer
5387 ;
5388
5388 integer : digit
5389 | integer digit
5390 ;
5391
5391 digit : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
5392 | 8 | 9 | A | B | C | D | E | F
5393 ;

```
- 5394 8. The value of a **NUMBER** token is interpreted as a numeral in the base specified by the  
 5395 value of the internal register **ibase** (described below). Each of the **digit** characters has the  
 5396 value from 0 to 15 in the order listed here, and the period character represents the radix  
 5397 point. The behaviour is undefined if digits greater than or equal to the value of **ibase**  
 5398 appear in the token. However, note the exception for single-digit values being assigned to  
 5399 **ibase** and **obase** themselves, in **Operations in bc** on page 150.
- 5400 9. The following keywords are recognised as tokens:
- |      |               |              |               |               |              |
|------|---------------|--------------|---------------|---------------|--------------|
| 5401 | <b>auto</b>   | <b>for</b>   | <b>length</b> | <b>return</b> | <b>sqrt</b>  |
| 5402 | <b>break</b>  | <b>ibase</b> | <b>obase</b>  | <b>scale</b>  | <b>while</b> |
| 5403 | <b>define</b> | <b>if</b>    | <b>quit</b>   |               |              |
- 5404 10. Any of the following characters occurring anywhere except within a keyword are  
 5405 recognised as the token **LETTER**:
- ```

5406   a b c d e f g h i j k l m n o p q r s t u v w x y z

```
- 5407 11. The following single-character and two-character sequences are recognised as the token
 5408 **ASSIGN_OP**:
- ```

5409 = += -= *= /= %= ^=

```

12. If an = character, as the beginning of a token, is followed by a – character with no intervening delimiter, the behaviour is undefined.

13. The following single-characters are recognised as the token **MUL\_OP**:

\* / %

14. The following single-character and two-character sequences are recognised as the token **REL\_OP**:

== <= >= != < >

15. The following two-character sequences are recognised as the token **INCR\_DECR**:

++ --

16. The following single characters are recognised as tokens whose names are the character:

<newline> ( ) , + - ; [ ] ^ { }

17. The token **EOF** will be returned when the end of input is reached.

### Operations in bc

There are three kinds of identifiers: ordinary identifiers, array identifiers and function identifiers. All three types consist of single lower-case letters. Array identifiers are followed by square brackets ([ ]). An array subscript is required except in an argument or auto list. Arrays are singly dimensioned and can contain up to {BC\_DIM\_MAX} elements. Indexing begins at zero so an array is indexed from 0 to {BC\_DIM\_MAX}–1. Subscripts will be truncated to integers. Function identifiers must be followed by parentheses, possibly enclosing arguments. The three types of identifiers do not conflict.

The following table summarises the rules for precedence and associativity of all operators. Operators on the same line have the same precedence; rows are in order of decreasing precedence.

| Operator                  | Associativity  |
|---------------------------|----------------|
| ++, --                    | not applicable |
| unary –                   | not applicable |
| ^                         | right to left  |
| *, /, %                   | left to right  |
| +, binary –               | left to right  |
| =, +=, -=, *=, /=, %=, ^= | right to left  |
| ==, <=, >=, !=, <, >      | none           |

**Table 3-3** Operators in *bc*

Each expression or named expression has a *scale*, which is the number of decimal digits that are maintained as the fractional portion of the expression.

*Named expressions* are places where values are stored. Named expressions are valid on the left side of an assignment. The value of a named expression is the value stored in the place named. Simple identifiers and array elements are named expressions; they have an initial value of zero and an initial scale of zero.

The internal registers **scale**, **ibase** and **obase** are all named expressions. The scale of an expression consisting of the name of one of these registers is zero; values assigned to any of these registers will be truncated to integers. The **scale** register contains a global value used in computing the scale of expressions (as described below). The value of the register **scale** is limited to  $0 \leq \text{scale} \leq \{\text{BC\_SCALE\_MAX}\}$  and has a default value of zero. The **ibase** and **obase** registers are the input and output number radix, respectively. The value of **ibase** is limited to:

$$2 \leq \text{ibase} \leq 16$$

The value of **obase** is limited to:

$$2 \leq \text{obase} \leq \{\text{BC\_BASE\_MAX}\}$$

When either **ibase** or **obase** is assigned a single **digit** value from the list in **Lexical Conventions** in **bc** on page 148, the value is assumed in hexadecimal. (For example, **ibase=A** sets to base ten, regardless of the current **ibase** value.) Otherwise, the behaviour is undefined when digits greater than or equal to the value of **ibase** appear in the input. Both **ibase** and **obase** have initial values of 10.

Internal computations will be conducted as if in decimal, regardless of the input and output bases, to the specified number of decimal digits. When an exact result is not achieved, (for example, **scale=0**;  $3.2/1$ ) the result will be truncated.

For all values of **obase** specified by this specification, numerical values will be output as follows:

1. If the value is less than zero, a hyphen (–) character will be output.
2. One of the following will be output, depending on the numerical value:
  - If the absolute value of the numerical value is greater than or equal to one, the integer portion of the value will be output as a series of digits appropriate to **obase** (as described below). The most significant non-zero digit will be output next, followed by each successively less significant digit.
  - If the absolute value of the numerical value is less than one but greater than zero and the scale of the numerical value is greater than zero, it is unspecified whether the character 0 is output.
  - If the numerical value is zero, the character 0 will be output.
3. If the scale of the value is greater than zero, a period character will be output, followed by a series of digits appropriate to **obase** (as described below) representing the most significant portion of the fractional part of the value. If *s* represents the scale of the value being output, the number of digits output will be *s* if **obase** is 10, less than or equal to *s* if **obase** is greater than 10, or greater than or equal to *s* if **obase** is less than 10. For **obase** values other than 10, this should be the number of digits needed to represent a precision of  $10^s$ .

For **obase** values from 2 to 16, valid digits are the first **obase** of the single characters:

0 1 2 3 4 5 6 7 8 9 A B C D E F

which represent the values zero to 15, inclusive, respectively.

5487 For bases greater than 16, each digit is written as a separate multi-digit decimal number. Each  
 5488 digit except the most significant fractional digit will be preceded a single space character. For  
 5489 bases from 17 to 100, *bc* will write two-digit decimal numbers; for bases from 101 to 1000, three-  
 5490 digit decimal strings and so on. For example, the decimal number 1024 in base 25 would be  
 5491 written as:

5492            $\Delta 01\Delta 15\Delta 24$

5493 in base 125, as:

5494            $\Delta 008\Delta 024$

5495 Very large numbers will be split across lines with 70 characters per line in the POSIX locale;  
 5496 other locales may split at different character boundaries. Lines that are continued must end with  
 5497 a backslash (\).

5498 A function call consists of a function name followed by parentheses containing a comma-  
 5499 separated list of expressions, which are the function arguments. A whole array passed as an  
 5500 argument is specified by the array name followed by empty square brackets. All function  
 5501 arguments are passed by value. As a result, changes made to the formal parameters have no  
 5502 effect on the actual arguments. If the function terminates by executing a **return** statement, the  
 5503 value of the function will be the value of the expression in the parentheses of the **return**  
 5504 statement or will be zero if no expression is provided or if there is no **return** statement.

5505 The result of **sqrt**(*expression*) will be the square root of the expression. The result will be  
 5506 truncated in the least significant decimal place. The scale of the result will be the scale of the  
 5507 expression or the value of **scale**, whichever is larger.

5508 The result of **length**(*expression*) will be the total number of significant decimal digits in the  
 5509 expression. The scale of the result will be zero.

5510 The result of **scale**(*expression*) will be the scale of the expression. The scale of the result will be  
 5511 zero.

5512 A numeric constant will be an expression. The scale will be the number of digits that follow the  
 5513 radix point in the input representing the constant, or zero if no radix point appears.

5514 The sequence ( *expression* ) will be an expression with the same value and scale as *expression*. The  
 5515 parentheses can be used to alter the normal precedence.

5516 The semantics of the unary and binary operators are as follows:

5517 **-expression**

5518           The result will be the negative of the *expression*. The scale of the result will be the scale  
 5519           of *expression*.

5520 The unary increment and decrement operators will not modify the scale of the named expression  
 5521 upon which they operate. The scale of the result will be the scale of that named expression.

5522 **++named-expression**

5523           The named expression will be incremented by one. The result will be the value of the  
 5524           named expression after incrementing.

5525 **--named-expression**

5526           The named expression will be decremented by one. The result will be the value of the  
 5527           named expression after decrementing.

5528 **named-expression++**

5529           The named expression will be incremented by one. The result will be the value of the  
 5530           named expression before incrementing.



5531 *named-expression* --  
 5532     The named expression will be decremented by one. The result will be the value of the  
 5533     named expression before decrementing.

5534     The exponentiation operator, circumflex (^), binds right to left.

5535 *expression* ^ *expression*  
 5536     The result will be the first *expression* raised to the power of the second *expression*. If the  
 5537     second expression is not an integer, the behaviour is undefined. If a is the scale of the  
 5538     left expression and b is the absolute value of the right expression, the scale of the result  
 5539     will be:

```
5540 if b >= 0 min(a * b, max(scale, a))
5541 if b < 0 scale
```

5542     The multiplicative operators ("\*", "/", "%") bind left to right. |

5543 *expression* \* *expression*  
 5544     The result will be the product of the two expressions. If a and b are the scales of the  
 5545     two expressions, then the scale of the result will be:

```
5546 min(a+b,max(scale,a,b))
```

5547 *expression* / *expression*  
 5548     The result will be the quotient of the two expressions. The scale of the result will be the  
 5549     value of **scale**.

5550 *expression* % *expression*  
 5551     For expressions a and b, a % b will be evaluated equivalent to the steps:  
 5552         1. Compute a/b to current scale.  
 5553         2. Use the result to compute:  
 5554             a - (a / b) \* b  
 5555         to scale:  
 5556             max(scale + scale(b), scale(a))

5557     The scale of the result will be:  
 5558             max(scale + scale(b), scale(a))

5559     When **scale** is zero, the "%" operator is the mathematical remainder operator. |

5560     The additive operators ("+", "-") bind left to right. |

5561 *expression* + *expression*  
 5562     The result will be the sum of the two expressions. The scale of the result will be the  
 5563     maximum of the scales of the expressions.

5564 *expression* - *expression*  
 5565     The result will be the difference of the two expressions. The scale of the result will be  
 5566     the maximum of the scales of the expressions.

5567     The assignment operators ("=", "+=", "-=", "\*=", "/=", "%=", "^=") bind right to left. |

5568 *named-expression* = *expression*  
 5569     This expression results in assigning the value of the expression on the right to the  
 5570     named expression on the left. The scale of both the named expression and the result  
 5571     will be the scale of *expression*.

5572 The compound assignment forms:

5573 *named-expression* <operator>= *expression*

5574 are equivalent to:

5575 *named-expression* = *named-expression* <operator> *expression*

5576 except that the *named-expression* will be evaluated only once.

5577 Unlike all other operators, the relational operators (" $<$ ", " $>$ ", " $<=$ ", " $>=$ ", " $==$ ", " $!=$ ") will be only  
5578 valid as the object of an **if**, **while** or inside a **for** statement.

5579 *expression1* < *expression2*

5580 The relation will be true if the value of *expression1* is strictly less than the value of  
5581 *expression2*.

5582 *expression1* > *expression2*

5583 The relation will be true if the value of *expression1* is strictly greater than the value of  
5584 *expression2*.

5585 *expression1* <= *expression2*

5586 The relation will be true if the value of *expression1* is less than or equal to the value of  
5587 *expression2*.

5588 *expression1* >= *expression2*

5589 The relation will be true if the value of *expression1* is greater than or equal to the value  
5590 of *expression2*.

5591 *expression1* == *expression2*

5592 The relation will be true if the values of *expression1* and *expression2* are equal.

5593 *expression1* != *expression2*

5594 The relation will be true if the values of *expression1* and *expression2* are unequal.

5595 There are only two storage classes in *bc*, global and automatic (local). Only identifiers that are to  
5596 be local to a function need be declared with the **auto** command. The arguments to a function  
5597 will be local to the function. All other identifiers are assumed to be global and available to all  
5598 functions. All identifiers, global and local, have initial values of zero. Identifiers declared as  
5599 auto will be allocated on entry to the function and released on returning from the function. They  
5600 therefore do not retain values between function calls. Auto arrays will be specified by the array  
5601 name followed by empty square brackets. On entry to a function, the old values of the names  
5602 that appear as parameters and as automatic variables are pushed onto a stack. Until the  
5603 function returns, reference to these names refers only to the new values.

5604 References to any of these names from other functions that are called from this function also  
5605 refer to the new value until one of those functions uses the same name for a local variable.

5606 When a statement is an expression, unless the main operator is an assignment, execution of the  
5607 statement will write the value of the expression followed by a newline character.

5608 When a statement is a string, execution of the statement will write the value of the string.

5609 Statements separated by semicolons or newline characters will be executed sequentially. In an  
5610 interactive invocation of *bc*, each time a newline character is read that satisfies the grammatical  
5611 production:

5612 *input\_item* : *semicolon\_list* NEWLINE

5613 the sequential list of statements making up the **semicolon\_list** will be executed immediately and  
5614 any output produced by that execution will be written without any delay due to buffering.

5615 In an **if** statement (**if** (*relation*) *statement*), the *statement* will be executed if the relation is true.

5616 The **while** statement (**while** (*relation*) *statement*) implements a loop in which the *relation* is tested;  
 5617 each time the *relation* is true, the *statement* will be executed and the *relation* retested. When the  
 5618 *relation* is false, execution will resume after *statement*.

5619 A **for** statement (**for** (*expression*; *relation*; *expression*) *statement*) is the same as:

```
5620 first-expression
5621 while (relation) {
5622 statement
5623 last-expression
5624 }
```

5625 All three expressions must be present.

5626 The **break** statement causes termination of a **for** or **while** statement.

5627 The **auto** statement (**auto** *identifier*[,*identifier*] ...) will cause the values of the identifiers to be  
 5628 pushed down. The identifiers can be ordinary identifiers or array identifiers. Array identifiers  
 5629 are specified by following the array name by empty square brackets. The **auto** statement must  
 5630 be the first statement in a function definition.

5631 A **define** statement:

```
5632 define LETTER (opt_parameter_list) {
5633 opt_auto_define_list
5634 statement_list
5635 }
```

5636 defines a function named *LETTER*. If a function named *LETTER* was previously defined, the  
 5637 **define** statement will replace the previous definition. The expression:

```
5638 LETTER (opt_argument_list)
```

5639 will invoke the function named *LETTER*. The behaviour is undefined if the number of  
 5640 arguments in the invocation does not match the number of parameters in the definition.  
 5641 Functions will be defined before they are invoked. A function will be considered to be defined  
 5642 within its own body, so recursive calls are valid. The values of numeric constants within a  
 5643 function will be interpreted in the base specified by the value of the **ibase** register when the  
 5644 function is invoked.

5645 The **return** statements (**return** and **return**(*expression*)) will cause termination of a function,  
 5646 popping of its auto variables and specifies the result of the function. The first form is equivalent  
 5647 to **return**(0). The value and scale of an invocation of the function will be the value and scale of  
 5648 the expression in parentheses.

5649 The **quit** statement (**quit**) will stop execution of a *bc* program at the point where the statement  
 5650 occurs in the input, even if it occurs in a function definition, or in an **if**, **for** or **while** statement.

5651 The following functions will be defined when the **-l** option is specified:

```
5652 s (expression)
5653 Sine of argument in radians.
```

```
5654 c (expression)
5655 Cosine of argument in radians.
```

```
5656 a (expression)
5657 Arctangent of argument.
```

5658 **l** ( *expression* )  
 5659       Natural logarithm of argument.

5660 **e** ( *expression* )  
 5661       Exponential function of argument.

5662 **j** ( *expression* , *expression* )  
 5663       Bessel function of integer order.

5664       The scale of an invocation of each of these functions will be the value of the **scale** register when  
 5665       the function is invoked. The behaviour is undefined if any of these functions is invoked with an  
 5666       argument outside the domain of the mathematical function.

#### 5667 **EXIT STATUS**

5668       The following exit values are returned:

5669       **0**               All input files were processed successfully.  
 5670       *unspecified*    An error occurred.

#### 5671 **CONSEQUENCES OF ERRORS**

5672       If any *file* operand is specified and the named file cannot be accessed, *bc* will write a diagnostic  
 5673       message to standard error and terminate without any further action.

5674       In an interactive invocation of *bc*, the utility should print an error message and recover following  
 5675       any error in the input. In a non-interactive invocation of *bc*, invalid input causes undefined  
 5676       behaviour.

#### 5677 **APPLICATION USAGE**

5678       Automatic variables in *bc* do not work in exactly the same way as in either C or PL/1.

5679       For historical reasons, the exit status from *bc* cannot be relied upon to indicate that an error has  
 5680       occurred. Returning zero after an error is possible. Therefore, *bc* should be used primarily by  
 5681       interactive users (who can react to error messages) or by application programs that can  
 5682       somehow validate the answers returned as not including error messages.

5683       The *bc* utility always uses the period (.) character to represent a radix point, regardless of any  
 5684       decimal-point character specified as part of the current locale. In languages like C or *awk*, the  
 5685       period character is used in program source, so it can be portable and unambiguous, while the  
 5686       locale-specific character is used in input and output. Because there is no distinction between  
 5687       source and input in *bc*, this arrangement would not be possible. Using the locale-specific  
 5688       character in *bc*'s input would introduce ambiguities into the language; consider the following  
 5689       example in a locale with a comma as the decimal-point character:

```
5690 define f(a,b) {
5691 ...
5692 }
5693 ...
5694 f(1,2,3)
```

5695       Because of such ambiguities, the period character is used in input. Having input follow different  
 5696       conventions from output would be confusing in either pipeline usage or interactive usage, so the  
 5697       period is also used in output.

#### 5698 **EXAMPLES**

5699       In the shell, the following assigns an approximation of the first ten digits of  $\pi$  to the variable *x*:

```
5700 x=$(printf "%s\n" 'scale = 10; 104348/33215' | bc)
```

5701 The following *bc* program prints the same approximation of  $\pi$ , with a label, to standard output:

```
5702 scale = 10
5703 "pi equals "
5704 104348 / 33215
```

5705 The following defines a function to compute an approximate value of the exponential function  
5706 (note that such a function is predefined if the `-l` option is specified):

```
5707 scale = 20
5708 define e(x){
5709 auto a, b, c, i, s
5710 a = 1
5711 b = 1
5712 s = 1
5713 for (i = 1; 1 == 1; i++){
5714 a = a*x
5715 b = b*i
5716 c = a/b
5717 if (c == 0) {
5718 return(s)
5719 }
5720 s = s+c
5721 }
5722 }
```

5723 The following prints approximate values of the exponential function of the first ten integers:

```
5724 for (i = 1; i <= 10; ++i) {
5725 e(i)
5726 }
```

#### 5727 FUTURE DIRECTIONS

5728 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
5729 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
5730 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
5731 finalised.

#### 5732 SEE ALSO

5733 *awk*.

#### 5734 CHANGE HISTORY

5735 First released in Issue 4.

#### 5736 Issue 5

5737 FUTURE DIRECTIONS section added.

## 5738 NAME

5739 bg — run jobs in the background

## 5740 SYNOPSIS

5741 JC bg [ *job\_id* ... ]

## 5742 DESCRIPTION

5743 If job control is enabled (see the description of *set -m*), the *bg* utility resumes suspended jobs  
 5744 from the current environment (see Section 2.12 on page 63) by running them as background jobs.  
 5745 If the job specified by *job\_id* is already a running background job, the *bg* utility has no effect and  
 5746 will exit successfully.

5747 Using *bg* to place a job into the background causes its process ID to become “known in the  
 5748 current shell execution environment”, as if it had been started as an asynchronous list; see  
 5749 Section 2.9.3 on page 50.

## 5750 OPTIONS

5751 None.

## 5752 OPERANDS

5753 The following operand is supported:

5754 *job\_id* Specify the job to be resumed as a background job. If no *job\_id* operand is given, the  
 5755 most recently suspended job is used. The format of *job\_id* is described in the entry for  
 5756 **job control job ID** in the XBD specification, **Chapter 2, Glossary**.

## 5757 STDIN

5758 Not used.

## 5759 INPUT FILES

5760 None.

## 5761 ENVIRONMENT VARIABLES

5762 The following environment variables affect the execution of *bg*:

5763 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 5764 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 5765 default locale will be used. If any of the internationalisation variables contains an  
 5766 invalid setting, the utility will behave as if none of the variables had been defined.

5767 *LC\_ALL*

5768 If set to a non-empty string value, override the values of all the other  
 5769 internationalisation variables.

5770 *LC\_CTYPE*

5771 Determine the locale for the interpretation of sequences of bytes of text data as  
 5772 characters (for example, single- as opposed to multi-byte characters in arguments).

5773 *LC\_MESSAGES*

5774 Determine the locale that should be used to affect the format and contents of diagnostic  
 5775 messages written to standard error.

5776 EX *NLSPATH*5777 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 5778 ASYNCHRONOUS EVENTS

5779 Default.

5780 **STDOUT**5781 The output of *bg* consists of a line in the format:

5782 "[%d] %s\n", &lt;job-number&gt;, &lt;command&gt;

5783 where the fields are as follows:

5784 &lt;job-number&gt;

5785 A number that can be used to identify the job to the *wait*, *fg* and *kill* utilities. Using  
5786 these utilities, the job can be identified by prefixing the job number with "%".

5787 &lt;command&gt;

5788 The associated command that was given to the shell.

5789 **STDERR**

5790 Used only for diagnostic messages.

5791 **OUTPUT FILES**

5792 None.

5793 **EXTENDED DESCRIPTION**

5794 None.

5795 **EXIT STATUS**

5796 The following exit values are returned:

5797 0 Successful completion.

5798 &gt;0 An error occurred.

5799 **CONSEQUENCES OF ERRORS**5800 If job control is disabled, the *bg* utility will exit with an error and no job will be placed in the  
5801 background.5802 **APPLICATION USAGE**5803 A job is generally suspended by typing the SUSP character (<control>-Z on most systems); see  
5804 the **XBD specification, Chapter 9, General Terminal Interface**. At that point, *bg* can put the job  
5805 into the background. This is most effective when the job is expecting no terminal input and its  
5806 output has been redirected to non-terminal files. A background job can be forced to stop when it  
5807 has terminal output by issuing the command:5808 `stty tostop`

5809 A background job can be stopped with the command:

5810 `kill -s stop job ID`5811 The *bg* utility will not work as expected when it is operating in its own utility execution  
5812 environment because that environment will have no suspended jobs. In the following examples:5813 `... | xargs bg`5814 `(bg)`5815 each *bg* operates in a different environment and will not share its parent shell's understanding of  
5816 jobs. For this reason, *bg* is generally implemented as a shell regular built-in.5817 **EXAMPLES**

5818 None.

5819 **FUTURE DIRECTIONS**

5820 None.

5821 **SEE ALSO**5822 *fg, kill, jobs, wait.*5823 **CHANGE HISTORY**

5824 First released in Issue 4.



5825 **NAME**5826 **c89** — compile standard C programs5827 **SYNOPSIS**

5828 **c89** [-c][-D *name*[=*value*]]...[-E][-g][-I *directory*] ... [-L *directory*]  
 5829 ... [-o *outfile*][-O][-s][-U *name*]... *operand* ...

5830 **DESCRIPTION**

5831 The **c89** utility is an interface to the standard C compilation system; it will accept source code  
 5832 conforming to the ISO C standard. The system conceptually consists of a compiler and link  
 5833 editor. The files referenced by *operands* will be compiled and linked to produce an executable  
 5834 file. (It is unspecified whether the linking occurs entirely within the operation of **c89**; some  
 5835 systems may produce objects that are not fully resolved until the file is executed.)

5836 If the **-c** option is specified, for all pathname operands of the form *file.c*, the files:

5837 `$(basename pathname .c).o`

5838 will be created as the result of successful compilation. If the **-c** option is not specified, it is  
 5839 unspecified whether such .o files are created or deleted for the *file.c* operands.

5840 If there are no options that prevent link editing (such as **-c** or **-E**), and all operands compile and  
 5841 link without error, the resulting executable file will be written according to the **-o** *outfile* option  
 5842 (if present) or to the file **a.out**.

5843 The executable file will be created as specified in the **XSH** specification, except that the file  
 5844 permissions will be set to:

5845 `S_IRWXO | S_IRWXG | S_IRWXU`

5846 and that the bits specified by the *umask* of the process will be cleared.

5847 **OPTIONS**

5848 The **c89** utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 5849 that:

- 5850 • The **-I** *library* operands have the format of options, but their position within a list of  
 5851 operands affects the order in which libraries are searched.
- 5852 • The order of specifying the **-I** and **-L** options is significant.
- 5853 • Portable applications must specify each option separately; that is, grouping option letters (for  
 5854 example, **-cO**) need not be recognised by all implementations.

5855 The following options are supported:

- 5856 **-c** Suppress the link-edit phase of the compilation, and do not remove any object files that  
 5857 are produced.
- 5858 **-g** Produce symbolic information in the object or executable files; the nature of this  
 5859 information is unspecified, and may be modified by implementation-dependent  
 5860 interactions with other options.
- 5861 **-s** Produce object or executable files, or both, from which symbolic and other information  
 5862 not required for proper execution using the **XSH** specification *exec* family has been  
 5863 removed (stripped). If both **-g** and **-s** options are present, the action taken is  
 5864 unspecified.

5865       **-o** *outfile*  
5866           Use the pathname *outfile*, instead of the default **a.out**, for the executable file produced.  
5867           If the **-o** option is present with **-c** or **-E**, the result is unspecified.

5868       **-D** *name*[=*value*]  
5869           Define *name* as if by a C-language **#define** directive. If no *=value* is given, a value of 1  
5870           will be used. The **-D** option has lower precedence than the **-U** option. That is, if *name*  
5871           is used in both a **-U** and a **-D** option, *name* will be undefined regardless of the order of  
5872           the options. Additional implementation-dependent *names* may be provided by the  
5873           compiler. Implementations support at least 2048 bytes of **-D** definitions and 256 *names*.

5874       **-E**       Copy C-language source files to standard output, expanding all preprocessor  
5875           directives; no compilation will be performed. If any operand is not a text file, the  
5876           effects are unspecified.

5877       **-I** *directory*  
5878           Change the algorithm for searching for headers whose names are not absolute  
5879           pathnames to look in the directory named by the *directory* pathname before looking in  
5880           the usual places. Thus, headers whose names are enclosed in double-quotes ("") will be  
5881           searched for first in the directory of the file with the **#include** line, then in directories  
5882           named in **-I** options, and last in the usual places. For headers whose names are  
5883           enclosed in angle brackets (<>), the header will be searched for only in directories  
5884           named in **-I** options and then in the usual places. Directories named in **-I** options will  
5885           be searched in the order specified. Implementations support at least ten instances of  
5886           this option in a single **c89** command invocation.

5887       **-L** *directory*  
5888           Change the algorithm of searching for the libraries named in the **-l** objects to look in  
5889           the directory named by the *directory* pathname before looking in the usual places.  
5890           Directories named in **-L** options will be searched in the order specified.  
5891           Implementations support at least ten instances of this option in a single **c89** command  
5892           invocation. If a directory specified by a **-L** option contains files named **libc.a**, **libm.a**,  
5893           **libl.a** or **liby.a**, the results are unspecified.

5894       **-O**       Optimise. The nature of the optimisation is unspecified.

5895       **-U** *name*  
5896           Remove any initial definition of *name*.

5897       Multiple instances of the **-D**, **-I**, **-U** and **-L** options can be specified.

5898   **OPERANDS**  
5899       An *operand* is either in the form of a pathname or the form **-l** *library*. At least one operand of the  
5900       pathname form must be specified. The following operands are supported:

5901       **file.c**     A C-language source file to be compiled and optionally linked. The operand must be of  
5902           this form if the **-c** option is used.

5903       **file.a**     A library of object files typically produced by the *ar* utility, and passed directly to the  
5904           link editor. Implementations may recognise implementation-dependent suffixes other  
5905           than .a as denoting object file libraries.

5906       **file.o**     An object file produced by **c89 -c** and passed directly to the link editor.  
5907           Implementations may recognise implementation-dependent suffixes other than .o as  
5908           denoting object files.

5909       The processing of other files is implementation-dependent.

5910        **-l library**  
 5911            (The letter ell.) Search the library named:  
 5912                liblibrary.a

5913        A library will be searched when its name is encountered, so the placement of a **-l**  
 5914        operand is significant. Several standard libraries can be specified in this manner, as  
 5915        described in the EXTENDED DESCRIPTION section. Implementations may recognise  
 5916        implementation-dependent suffixes other than .a as denoting libraries.

5917   **STDIN**  
 5918        Not used.

5919   **INPUT FILES**  
 5920        The input file must be one of the following: a text file containing a C-language source program;  
 5921        an object file in the format produced by **c89 -c** or a library of object files, in the format produced  
 5922        by archiving zero or more object files, using **ar**. Implementations may supply additional utilities  
 5923        that produce files in these formats. Additional input file formats are implementation-  
 5924        dependent.

5925   **ENVIRONMENT VARIABLES**  
 5926        The following environment variables affect the execution of **c89**:

5927        **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 5928        **LANG** is unset or null, the corresponding value from the implementation-dependent  
 5929        default locale will be used. If any of the internationalisation variables contains an  
 5930        invalid setting, the utility will behave as if none of the variables had been defined.

5931        **LC\_ALL**  
 5932            If set to a non-empty string value, override the values of all the other  
 5933            internationalisation variables.

5934        **LC\_CTYPE**  
 5935            Determine the locale for the interpretation of sequences of bytes of text data as  
 5936            characters (for example, single- as opposed to multi-byte characters in arguments and  
 5937            input files).

5938        **LC\_MESSAGES**  
 5939            Determine the locale that should be used to affect the format and contents of diagnostic  
 5940            messages written to standard error.

5941   EX       **NLSPATH**  
 5942            Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

5943        **TMPDIR**  
 5944        EX       Provide a pathname that will override the default directory for temporary files, if any.

5945   **ASYNCHRONOUS EVENTS**  
 5946        Default.

5947   **STDOUT**  
 5948        If more than one file operand ending in .c (or possibly other unspecified suffixes) is given, for  
 5949        each such file:  
 5950            "%s:\n", <file>  
 5951        may be written. These messages, if written, will precede the processing of each input file; they  
 5952        will not be written to the standard output if they are written to the standard error, as described  
 5953        in the STDERR section.

5954 If the **-E** option is specified, the standard output will be a text file that represents the results of  
 5955 the preprocessing stage of the language; it may contain extra information appropriate for  
 5956 subsequent compilation passes.

#### 5957 **STDERR**

5958 Used only for diagnostic messages. If more than one file operand ending in **.c** (or possibly other  
 5959 unspecified suffixes) is given, for each such file:

5960 `"%s:\n", <file>`

5961 may be written to allow identification of the diagnostic and warning messages with the  
 5962 appropriate input file. These messages, if written, will precede the processing of each input file;  
 5963 they will not be written to the standard error if they are written to the standard output, as  
 5964 described in the **STDOUT** section.

5965 This utility may produce warning messages about certain conditions that do not warrant  
 5966 returning an error (non-zero) exit value.

#### 5967 **OUTPUT FILES**

5968 Object files or executable files or both are produced in unspecified formats.

#### 5969 **EXTENDED DESCRIPTION**

##### 5970 **Standard Libraries**

5971 The **c89** utility recognises the following **-l** operands for standard libraries:

5972 **-l c** This operand makes visible all library functions referenced in the **XSH** specification  
 5973 with the possible exception of those functions listed as residing in **<aio.h>**, **<math.h>**,  
 5974 **<mqueue.h>**, **<pthread.h>**, **<sched.h>**, **<semaphore.h>**, **pthread\_atfork()** in **<unistd.h>**  
 5975 and those functions marked as an RT extension in **<sys/mman.h>** and **<time.h>**. This  
 5976 operand is not required to be present to cause a search of this library.

5977 **-l l** This operand makes visible all functions required by the C-language output of *lex* that  
 5978 are not made available through the **-l c** operand.

##### 5979 **-l pthread**

5980 This operand makes visible all functions referenced in **<pthread.h>** and **pthread\_atfork()**  
 5981 referenced in **<unistd.h>**. An implementation may search this library in the absence of  
 5982 this operand.

5983 **-l m** This operand makes visible all functions referenced in **<math.h>**. An implementation  
 5984 may search this library in the absence of this operand.

5985 **-l rt** This operand makes visible all functions referenced in **<aio.h>**, **<mqueue.h>**,  
 5986 **<sched.h>** and **<semaphore.h>**, and those functions marked as an RT extension in  
 5987 **<sys/mman.h>** and **<time.h>**. An implementation may search this library in the  
 5988 absence of this operand.

5989 **-l y** This operand makes visible all functions required by the C-language output of *yacc* that  
 5990 are not made available through the **-l c** operand.

5991 In the absence of options that inhibit invocation of the link editor, such as **-c** or **-E**, the **c89** utility  
 5992 will cause the equivalent of a **-l c** operand to be passed to the link editor as the last **-l** operand,  
 5993 causing it to be searched after all other object files and libraries are loaded.

5994 It is unspecified whether the libraries **libc.a**, **libm.a**, **librt.a**, **libpthread.a**, **libl.a** or **liby.a** exist as  
 5995 regular files. The implementation may accept as **-l** operands names of objects that do not exist  
 5996 as regular files.

**External Symbols**

The C compiler and link editor support the significance of external symbols up to a length of at least 31 bytes; the action taken upon encountering symbols exceeding the implementation-dependent maximum symbol length is unspecified.

The compiler and link editor support a minimum of 511 external symbols per source or object file, and a minimum of 4095 external symbols in total. A diagnostic message will be written to the standard output if the implementation-dependent limit is exceeded; other actions are unspecified.

**Programming Environments**

All implementations will support one of the following programming environments as a default. Implementations may support more than one of the following programming environments. Applications can use *sysconf()* or *getconf* to determine which programming environments are supported.

| Programming Environment<br><i>getconf</i> Name | Bits in<br>int | Bits in<br>long | Bits in<br>pointer | Bits in<br>off_t |
|------------------------------------------------|----------------|-----------------|--------------------|------------------|
| XBS5_ILP32_OFF32                               | 32             | 32              | 32                 | 32               |
| XBS5_ILP32_OFFBIG                              | 32             | 32              | 32                 | ≥64              |
| XBS5_LP64_OFF64                                | 32             | 64              | 64                 | 64               |
| XBS5_LP64_OFFBIG                               | ≥32            | ≥64             | ≥64                | ≥64              |

**Table 3-4** Programming Environments — Type Sizes

Implementations provide configuration strings for C compiler flags, linker/loader flags and libraries for each supported environment. When an application needs to use a specific programming environment rather than the implementation default programming environment while compiling, the application must first verify that the implementation supports the desired environment. If the desired programming environment is supported, the application must then invoke *c89* with the appropriate C compiler flags as the first options for the compile, the appropriate linker/loader flags after any other options but before any operands, and the appropriate libraries at the end of the operands.

Portable applications must not attempt to link together object files compiled for different programming models. Applications must also be aware that binary data placed in shared memory or in files might not be recognised by applications built for other programming models.

| Programming Environment<br><i>getconf</i> Name | Use                                                  | c89 and cc Arguments<br><i>getconf</i> Name                                     |
|------------------------------------------------|------------------------------------------------------|---------------------------------------------------------------------------------|
| XBS5_ILP32_OFF32                               | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | XBS5_ILP32_OFF32_CFLAGS<br>XBS5_ILP32_OFF32_LDFLAGS<br>XBS5_ILP32_OFF32_LIBS    |
| XBS5_ILP32_OFFBIG                              | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | XBS5_ILP32_OFFBIG_CFLAGS<br>XBS5_ILP32_OFFBIG_LDFLAGS<br>XBS5_ILP32_OFFBIG_LIBS |
| XBS5_LP64_OFF64                                | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | XBS5_LP64_OFF64_CFLAGS<br>XBS5_LP64_OFF64_LDFLAGS<br>XBS5_LP64_OFF64_LIBS       |
| XBS5_LPBIG_OFFBIG                              | C Compiler Flags<br>Linker/Loader Flags<br>Libraries | XBS5_LPBIG_OFFBIG_CFLAGS<br>XBS5_LPBIG_OFFBIG_LDFLAGS<br>XBS5_LPBIG_OFFBIG_LIBS |

**Table 3-5** Programming Environments — c89 and cc Arguments

## EXIT STATUS

The following exit values are returned:

- 0 Successful compilation or link edit.
- >0 An error occurred.

## CONSEQUENCES OF ERRORS

When *c89* encounters a compilation error that causes an object file not to be created, it will write a diagnostic to standard error and continue to compile other source code operands, but it will not perform the link phase and will return a non-zero exit status. If the link edit is unsuccessful, a diagnostic message will be written to standard error and *c89* will exit with a non-zero status. A portable application must rely on the exit status of *c89*, rather than on the existence or mode of the executable file.

## APPLICATION USAGE

Since the *c89* utility usually creates files in the current directory during the compilation process, it is typically necessary to run the *c89* utility in a directory in which a file can be created.

On systems conforming to the ISO/IEC 9945-2: 1993 standard, *c89* may be provided only as part of the C-Language Development Option; XSI-conformant systems always provide *c89*.

Some historical implementations have created *.o* files when *-c* is not specified and more than one source file is given. Since this area is left unspecified, the application cannot rely on *.o* files being created, but it also must be prepared for any related *.o* files that already exist being deleted at the completion of the link edit.

Some historical implementations have permitted *-L* options to be interspersed with *-l* operands on the command line. For an application to compile consistently on systems that do not behave like this, it is necessary for a portable application to supply all *-L* options before any of the *-l* options.

There is the possible implication that if a user supplies versions of the standard library functions (before they would be encountered by an implicit *-l c* or explicit *-l m*), that those versions would be used in place of the standard versions. There are various reasons this might not be true (functions defined as macros, manipulations for clean name space, and so on), so the existence of files named in the same manner as the standard libraries within the *-L* directories is explicitly stated to produce unspecified behaviour.

6075 All of the interfaces specified in the **XSH** specification may be made visible by implementations  
 6076 when the Standard C Library is searched. Portable applications must explicitly request  
 6077 searching the other standard libraries when functions made visible by those libraries are used.

6078 An application strictly portable to the ISO/IEC 9945-2:1993 standard cannot rely on *TMPDIR*  
 6079 overriding the default temporary directory. On XSI-conformant systems, however, this will  
 6080 always be the case.

#### 6081 EXAMPLES

6082 The following are examples of usage:

6083 `c89 -o foo foo.c`

6084 Compiles **foo.c** and creates the executable file **foo**.

6085 `c89 -c foo.c`

6086 Compiles **foo.c** and creates the object file **foo.o**.

6087 `c89 foo.c`

6088 Compiles **foo.c** and creates the executable file **a.out**.

6089 `c89 foo.c bar.o`

6090 Compiles **foo.c**, links it with **bar.o**, and creates the executable file **a.out**. Also creates  
 6091 and leaves **foo.o**.

6092 The following example shows how an application using threads interfaces can test for support  
 6093 of and use a programming environment supporting 32-bit **int**, **long** and **pointer** types and an  
 6094 **off\_t** type using at least 64 bits:

```
6095 if [$(getconf XBS5_ILP32_OFFBIG) != "-1"]
6096 then
6097 c89 $(getconf XBS5_ILP32_OFFBIG_CFLAGS) -D_XOPEN_SOURCE=500 \
6098 $(getconf XBS5_ILP32_OFFBIG_LDFLAGS) foo.c -o foo \
6099 -l pthread $(getconf XBS5_ILP32_OFFBIG_LIBS)
6100 else
6101 echo ILP32_OFFBIG programming environment not supported
6102 exit 1
6103 fi
```

6104 The following examples clarify the use and interactions of **-L** options and **-l** operands:

6105 1. Consider the case in which module **a.c** calls function *f()* in library **libQ.a**, and module **b.c**  
 6106 calls function *g()* in library **libp.a**. Assume that both libraries reside in **/a/b/c**. The  
 6107 command line to compile and link in the desired way is:

6108 `c89 -L /a/b/c main.o a.c -l Q b.c -l p`

6109 In this case the **-l Q** operand need only precede the first **-l p** operand, since both **libQ.a**  
 6110 and **libp.a** reside in the same directory.

6111 2. Multiple **-L** operands can be used when library name collisions occur. Building on the  
 6112 previous example, suppose that the user wants to use a new **libp.a**, in **/a/a/a**, but still wants  
 6113 *f()* from **/a/b/c/libQ.a**:

6114 `c89 -L /a/a/a -L /a/b/c main.o a.c -l Q b.c -l p`

6115 In this example, the linker searches the **-L** options in the order specified, and finds  
 6116 **/a/a/a/libp.a** before **/a/b/c/libp.a** when resolving references for **b.c**. The order of the **-l**  
 6117 operands is still important, however.

6118 **FUTURE DIRECTIONS**

6119       None.

6120 **SEE ALSO**6121       *ar*, *cc*, *getconf*, *make*, *nm*, *strip*, the **XSH** specification description of *sysconf()*, *umask*. |6122 **CHANGE HISTORY**

6123       First released in Issue 4.

6124 **Issue 4, Version 2**6125       In the **Standard Libraries** subsection, the **-l c** operand describes access to traditional interfaces  
6126       if **\_XOPEN\_UNIX** is defined. |6127 **Issue 5**6128       In the **Standard Libraries** subsection, the **-l pthread** and **-l rt** operands are added. |6129       A section on the use of *sysconf()* and *getconf* for determining the supported programming  
6130       environments is added. |



6131 **NAME**

6132 cal — print a calendar

6133 **SYNOPSIS**6134 EX cal `[[month] year ]`6135 **DESCRIPTION**

6136 The *cal* utility writes a Gregorian calendar to standard output. If the *year* operand is specified, a  
 6137 calendar for that year is written. If no operands are specified, a calendar for the current month is  
 6138 written.

6139 **OPTIONS**

6140 None.

6141 **OPERANDS**

6142 The following operands are supported:

6143 *month* Specify the month to be displayed, represented as a decimal integer from 1 (January) to  
 6144 12 (December). The default is the current month.

6145 *year* Specify the year for which the calendar is displayed, represented as a decimal integer  
 6146 from 1 to 9999. The default is the current year.

6147 **STDIN**

6148 Not used.

6149 **INPUT FILES**

6150 None.

6151 **ENVIRONMENT VARIABLES**6152 The following environment variables affect the execution of *cal*:

6153 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 6154 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 6155 default locale will be used. If any of the internationalisation variables contains an  
 6156 invalid setting, the utility will behave as if none of the variables had been defined.

6157 *LC\_ALL*

6158 If set to a non-empty string value, override the values of all the other  
 6159 internationalisation variables.

6160 *LC\_CTYPE*

6161 Determine the locale for the interpretation of sequences of bytes of text data as  
 6162 characters (for example, single- as opposed to multi-byte characters in arguments).

6163 *LC\_MESSAGES*

6164 Determine the locale that should be used to affect the format and contents of diagnostic  
 6165 messages written to standard error, and informative messages written to standard  
 6166 output.

6167 *LC\_TIME*

6168 Determine the format and contents of the calendar.

6169 *NLSPATH*6170 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.6171 *TZ* Determine the timezone used to calculate the value of the current month.6172 **ASYNCHRONOUS EVENTS**

6173 Default.

**6174 STDOUT**

6175           The standard output is used to display the calendar, in an unspecified format.

**6176 STDERR**

6177           Used only for diagnostic messages.

**6178 OUTPUT FILES**

6179           None.

**6180 EXTENDED DESCRIPTION**

6181           None.

**6182 EXIT STATUS**

6183           The following exit values are returned:

6184           0   Successful completion.

6185           >0  An error occurred.

**6186 CONSEQUENCES OF ERRORS**

6187           Default.

**6188 APPLICATION USAGE**

6189           Note that:

6190           cal 83

6191           refers to A.D. 83, not 1983.

**6192 EXAMPLES**

6193           None.

**6194 FUTURE DIRECTIONS**

6195           None.

**6196 SEE ALSO**

6197           None.

**6198 CHANGE HISTORY**

6199           First released in Issue 2.

**6200 Issue 4**

6201           Format reorganised.

6202           Internationalised environment variable support mandated.

6203 **NAME**  
 6204 calendar — reminder service (**LEGACY**)

6205 **SYNOPSIS**  
 6206 EX `calendar`

6207 **DESCRIPTION**  
 6208 The *calendar* utility consults the file **calendar** in the current directory and writes lines that  
 6209 contain today's or tomorrow's date anywhere in the line to standard output. On Fridays and  
 6210 weekends, *tomorrow* extends to the following Monday, inclusive.

6211 **OPTIONS**  
 6212 None.

6213 **OPERANDS**  
 6214 None.

6215 **STDIN**  
 6216 Not used.

6217 **INPUT FILES**  
 6218 The **calendar** file in the current directory is a text file. Each line can contain text that includes a  
 6219 string, in any location, that is interpreted as *today's* or *tomorrow's* date. Month-day date formats  
 6220 such as *Aug. 24*, *august 24* and *8/24* are recognised.

6221 **ENVIRONMENT VARIABLES**  
 6222 The following environment variables may affect the execution of *calendar*:

6223 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 6224 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 6225 default locale will be used. If any of the internationalisation variables contains an  
 6226 invalid setting, the utility will behave as if none of the variables had been defined.

6227 **LC\_ALL**  
 6228 If set to a non-empty string value, override the values of all the other  
 6229 internationalisation variables.

6230 **LC\_CTYPE**  
 6231 Determine the locale for the interpretation of sequences of bytes of text data as  
 6232 characters (for example, single- as opposed to multi-byte characters in input files).

6233 **LC\_MESSAGES**  
 6234 Determine the locale that should be used to affect the format and contents of diagnostic  
 6235 messages written to standard error.

6236 **LC\_TIME**  
 6237 Determine the format of the date strings recognised by the *calendar* utility.

6238 **NLSPATH**  
 6239 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

6240 **TZ** Determine the timezone used to qualify the date strings recognised by the *calendar*  
 6241 utility.

6242 **ASYNCHRONOUS EVENTS**  
 6243 Default.

6244 **STDOUT**  
 6245 The standard output contains all of the selected lines from the **calendar** file.

6246 **STDERR**

6247 Not used.

6248 **OUTPUT FILES**

6249 None.

6250 **EXTENDED DESCRIPTION**

6251 None.

6252 **EXIT STATUS**

6253 The following exit values are returned:

6254 0 Successful completion.

6255 &gt;0 An error occurred.

6256 **CONSEQUENCES OF ERRORS**

6257 Default.

6258 **APPLICATION USAGE**6259 Some implementations support extensions that use operands. Portable applications must not  
6260 use any operands to *calendar*.6261 **EXAMPLES**

6262 None.

6263 **FUTURE DIRECTIONS**

6264 None.

6265 **SEE ALSO**

6266 None.

6267 **CHANGE HISTORY**

6268 First released in Issue 2.

6269 **Issue 4**

6270 Format reorganised.

6271 Internationalised environment variable support made optional.

6272 Marked TO BE WITHDRAWN.

6273 **Issue 5**

6274 Marked LEGACY.

6275 **NAME**6276           cancel — cancel printer requests (**LEGACY**)6277 **SYNOPSIS**6278 UN EX   cancel [*ID ...*] *printer ...*6279 UN EX   cancel *ID ...*[*printer ...*]6280 **DESCRIPTION**

6281           The *cancel* utility cancels printer requests that were made by an *lp* command. The cancellation of  
 6282           a request that is currently printing frees the printer to print its next available request.

6283           Cancelling requests from other users requires appropriate privileges. For each request  
 6284           successfully cancelled by a user who did not submit the request, the submitter may be notified  
 6285           that the request was cancelled.

6286           The *cancel* utility cannot reliably cancel print requests in all conceivable circumstances. When  
 6287           the printer is under the control of another operating system or resides on a remote system across  
 6288           a network, it might not be possible to affect the status of the print job after it has left the control  
 6289           of the local operating system. Even on local printers, spooling hardware in the printer may  
 6290           make it appear that the print job has been completed long before the final page is printed.

6291 **OPTIONS**

6292           None.

6293 **OPERANDS**

6294           The following operands are supported:

6295           *ID*        A request *ID*, as returned by *lp*. Specifying a *request ID* cancels the associated request  
 6296           even if it is currently printing.

6297           *printer*   A printer name (for a complete list of printer names, use *lpstat*). Specifying a printer  
 6298           cancels the request that is currently printing on that printer.

6299 **STDIN**

6300           Not used.

6301 **INPUT FILES**

6302           None.

6303 **ENVIRONMENT VARIABLES**6304           The following environment variables affect the execution of *cancel*:

6305           *LANG*    Provide a default value for the internationalisation variables that are unset or null. If  
 6306           *LANG* is unset or null, the corresponding value from the implementation-dependent  
 6307           default locale will be used. If any of the internationalisation variables contains an  
 6308           invalid setting, the utility will behave as if none of the variables had been defined.

6309           *LC\_ALL*

6310           If set to a non-empty string value, override the values of all the other  
 6311           internationalisation variables.

6312           *LC\_CTYPE*

6313           Determine the locale for the interpretation of sequences of bytes of text data as  
 6314           characters (for example, single- as opposed to multi-byte characters in arguments).

6315           *LC\_MESSAGES*

6316           Determine the locale that should be used to affect the format and contents of diagnostic  
 6317           messages written to standard error, and informative messages written to standard  
 6318           output.

6319            *NLSPATH*  
6320            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

6321 **ASYNCHRONOUS EVENTS**  
6322            Default.

6323 **STDOUT**  
6324            The standard output is a text file containing the status of each cancellation request, in an  
6325            unspecified format.

6326 **STDERR**  
6327            Used only for diagnostic messages.

6328 **OUTPUT FILES**  
6329            If mail notification is used to inform users of their requests being cancelled by other users, mail  
6330            files will be modified.

6331 **EXTENDED DESCRIPTION**  
6332            None.

6333 **EXIT STATUS**  
6334            The following exit values are returned:  
6335            0    Successful completion.  
6336            >0   An error occurred.

6337 **CONSEQUENCES OF ERRORS**  
6338            Default.

6339 **APPLICATION USAGE**  
6340            None.

6341 **EXAMPLES**  
6342            None.

6343 **FUTURE DIRECTIONS**  
6344            None.

6345 **SEE ALSO**  
6346            *lp, lpstat, mailx*.

6347 **CHANGE HISTORY**  
6348            First released in Issue 2.

6349 **Issue 4**  
6350            Format reorganised and separated from the *lp* description.  
6351            Internationalised environment variable support mandated.

6352 **Issue 5**  
6353            Marked LEGACY.

6354 **NAME**

6355       cat — concatenate and print files

6356 **SYNOPSIS**6357       cat [-u][*file* ...]6358 **DESCRIPTION**6359       The *cat* utility reads files in sequence and writes their contents to the standard output in the  
6360       same sequence.6361 **OPTIONS**6362       The *cat* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

6363       The following option is supported:

6364       **-u**       Write bytes from the input file to the standard output without delay as each is read.6365 **OPERANDS**

6366       The following operand is supported:

6367       *file*       A pathname of an input file. If no *file* operands are specified, the standard input is  
6368       used. If a *file* is "-", the *cat* utility will read from the standard input at that point in the  
6369       sequence. The *cat* utility will not close and reopen standard input when it is referenced  
6370       in this way, but will accept multiple occurrences of "-" as a *file* operand.6371 **STDIN**6372       The standard input is used only if no *file* operands are specified, or if a *file* operand is "-". See the  
6373       INPUT FILES section.6374 **INPUT FILES**

6375       The input files can be any file type.

6376 **ENVIRONMENT VARIABLES**6377       The following environment variables affect the execution of *cat*:6378       **LANG**       Provide a default value for the internationalisation variables that are unset or null. If  
6379       **LANG** is unset or null, the corresponding value from the implementation-dependent  
6380       default locale will be used. If any of the internationalisation variables contains an  
6381       invalid setting, the utility will behave as if none of the variables had been defined.6382       **LC\_ALL**6383       If set to a non-empty string value, override the values of all the other  
6384       internationalisation variables.6385       **LC\_CTYPE**6386       Determine the locale for the interpretation of sequences of bytes of text data as  
6387       characters (for example, single- as opposed to multi-byte characters in arguments).6388       **LC\_MESSAGES**6389       Determine the locale that should be used to affect the format and contents of diagnostic  
6390       messages written to standard error.6391   EX       **NLSPATH**6392       Determine the location of message catalogues for the processing of **LC\_MESSAGES**.6393 **ASYNCHRONOUS EVENTS**

6394       Default.

6395 **STDOUT**6396       The standard output will contain the sequence of bytes read from the input files. Nothing else  
6397       will be written to the standard output.

6398 **STDERR**

6399       Used only for diagnostic messages.

6400 **OUTPUT FILES**

6401       None.

6402 **EXTENDED DESCRIPTION**

6403       None.

6404 **EXIT STATUS**

6405       The following exit values are returned:

6406           0   All input files were output successfully.

6407           &gt;0  An error occurred.

6408 **CONSEQUENCES OF ERRORS**

6409       Default.

6410 **APPLICATION USAGE**6411       The **-u** option has value in prototyping non-blocking reads from FIFOs. The intent is to support  
6412       the following sequence:

```
6413 mkfifo foo
6414 cat -u foo > /dev/tty13 &
6415 cat -u > foo
```

6416       It is unspecified whether standard output is or is not buffered in the default case. This is  
6417       sometimes of interest when standard output is associated with a terminal, since buffering may  
6418       delay the output. The presence of the **-u** option guarantees that unbuffered I/O is available. It  
6419       is implementation-dependent whether the *cat* utility buffers output if the **-u** option is not  
6420       specified. Traditionally, the **-u** option is implemented using the equivalent of the **XSH**  
6421       specification *setvbuf()* function.

6422 **EXAMPLES**

6423       The following command:

6424       

```
cat myfile
```

6425       writes the contents of the file **myfile** to standard output.

6426       The following command:

6427       

```
cat doc1 doc2 > doc.all
```

6428       concatenates the files **doc1** and **doc2** and writes the result to **doc.all**.

6429       Because of the shell language mechanism used to perform output redirection, a command such  
6430       as this:

6431       

```
cat doc doc.end > doc
```

6432       causes the original data in **doc** to be lost.

6433       The command:

6434       

```
cat start - middle - end > file
```

6435       when standard input is a terminal, gets two arbitrary pieces of input from the terminal with a  
6436       single invocation of *cat*. Note, however, that if standard input is a regular file, this would be  
6437       equivalent to the command:

6438       

```
cat start - middle /dev/null end > file
```



6439 because the entire contents of the file would be consumed by *cat* the first time "-" was used as a  
6440 *file* operand and an end-of-file condition would be detected immediately when "-" was  
6441 referenced the second time.

6442 **FUTURE DIRECTIONS**

6443 None.

6444 **SEE ALSO**

6445 *more*.

6446 **CHANGE HISTORY**

6447 First released in Issue 2.

6448 **Issue 4**

6449 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 6450 NAME

6451 cc — a C-language compilation system (**LEGACY**)

## 6452 SYNOPSIS

```
6453 EX cc [-c][-C][-e epsym] [-D name[=value]]... [-E][-f][-F][-g]
6454 [-I directory]... [-L directory]... [-o outfile][-O][-p][-P]
6455 [-q][-r][-s][-S][-u symname]... [-U name]... [-W options]... operand...
```

## 6457 DESCRIPTION

6458 The *cc* utility is an interface to an unspecified C-language compilation system. The system  
 6459 conceptually consists of a preprocessor, compiler, optimiser, assembler and link editor. The *cc*  
 6460 utility processes the supplied options and then executes the various tools with the appropriate  
 6461 arguments.

6462 The suffix of the pathname versions of an *operand* indicates how it is to be treated. See the  
 6463 OPERANDS section.

6464 The files referenced by *operands* will be compiled/assembled and linked to produce an  
 6465 executable file. (It is unspecified whether the linking occurs entirely within the operation of *cc*;  
 6466 some systems may produce objects that are not fully resolved until the file is executed.)

6467 If the *-c* option is specified, for all pathname operands of the form *file.c*, the files:

6468     \$(basename *pathname* .c).o

6469 will be created as the result of successful compilation. Similar results occur for pathname  
 6470 operands of the form *file.i* and *.s*. If the *-c* option is not specified, it is unspecified whether such  
 6471 *.o* files are created or deleted for these operands.

6472 If there are no options that prevent link editing (such as *-c* or *-E*), and all operands compile and  
 6473 link without error, the resulting executable file will be written according to the *-o outfile* option  
 6474 (if present) or to the file *a.out*.

6475 The executable file will be created as specified in the **XSH** specification, except that the file  
 6476 permissions will be set to:

6477     S\_IRWXO | S\_IRWXG | S\_IRWXU

6478 and that the bits specified by the *umask* of the process will be cleared.

## 6479 OPTIONS

6480 The *cc* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 6481 that:

- 6482 • The *-I library* operands have the format of options, but their position within a list of  
 6483 operands affects the order in which libraries are searched.
- 6484 • The order of specifying the *-I* and *-L* options is significant.
- 6485 • Portable applications must specify each option separately; that is, grouping option letters (for  
 6486 example, *-cO*) need not be recognised by all implementations.

6487 The following options are supported:

6488 *-c*     Suppress the link-edit phase of the compilation, and do not remove any object files that  
 6489 are produced.

|      |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
|------|-------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 6490 | UN    | <b>-E</b>                 | Run only the preprocessor on the named C-language programs and send the result to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |
| 6491 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6492 |       | <b>-f</b>                 | Include floating-point support for systems without an automatically included floating point implementation. This option is ignored on systems that do not need it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |
| 6493 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6494 | PI    | <b>-F</b>                 | This option is reserved for implementation-dependent optimisation directives.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 6495 | PI OP | <b>-g</b>                 | Cause the compiler to generate additional information needed for use by a debugger (possibly <i>sdb</i> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 6496 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6497 |       | <b>-o outfile</b>         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6498 |       |                           | Use the name <i>outfile</i> instead of the default <b>a.out</b> for the executable file produced. This is a link-edit option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 6499 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6500 |       | <b>-O</b>                 | Do compilation phase optimisation. This option will not affect <i>.s</i> files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6501 | PI OP | <b>-p</b>                 | This option is reserved for invoking implementation-dependent profiling procedures.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 6502 | UN    | <b>-P</b>                 | Run only the preprocessor on the named C-language programs and leave the result on corresponding files suffixed <i>.i</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |  |
| 6503 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6504 | PI    | <b>-q</b>                 | This option is reserved for specifying implementation-dependent profiling directives.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |  |
| 6505 | UN    | <b>-S</b>                 | Compile and do not assemble the named C-language programs, and leave the assembler-language output on corresponding files suffixed <i>.s</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |  |
| 6506 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6507 | PI    | <b>-W c,arg[,arg ...]</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6508 |       |                           | Pass the arguments <i>arg</i> to phase <i>c</i> where <i>c</i> is one of [p02al] indicating preprocessing, compiling, optimising, assembling or link editing phases, respectively. For example, <b>-Wa,-m</b> passes <b>-m</b> to the assembler phase.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |
| 6509 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6510 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6511 |       |                           | The <i>cc</i> utility also recognises a number of options that it will pass (with their associated arguments) directly to another phase of the <i>cc</i> utility. The use of the <b>-W</b> option is not required for these options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 6512 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6513 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6514 |       |                           | The following options are passed by <i>cc</i> (with their associated arguments) to the preprocessor phase:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 6515 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6516 |       | <b>-C</b>                 | By default, the preprocessor strips C-language style comments. If the <b>-C</b> option is specified, all comments (except those found on preprocessor directive lines) are passed along.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 6517 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6518 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6519 |       | <b>-D name[=value]</b>    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6520 |       |                           | Define <i>name</i> as if by a C-language <b>#define</b> directive. If no <i>=value</i> is given, a value of 1 will be used. The <b>-D</b> option has lower precedence than the <b>-U</b> option. That is, if <i>name</i> is used in both a <b>-U</b> and a <b>-D</b> option, <i>name</i> will be undefined regardless of the order of the options. Additional implementation-dependent <i>names</i> may be provided by the compiler. Implementations support at least 2048 bytes of <b>-D</b> definitions and 256 <i>names</i> .                                                                                                                                                                                                                                                              |  |
| 6521 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6522 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6523 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6524 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6525 |       | <b>-I directory</b>       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6526 |       |                           | Change the algorithm for searching for headers whose names are not absolute pathnames to look in the directory named by the <i>directory</i> pathname before looking in the usual places. Thus, headers whose names are enclosed in double-quotes ("") will be searched for first in the directory of the file with the <b>#include</b> line, then in directories named in <b>-I</b> options, and last in the usual places. For headers whose names are enclosed in angle brackets (<>), the header will be searched for only in directories named in <b>-I</b> options and then in the usual places. Directories named in <b>-I</b> options will be searched in the order specified. Implementations support at least ten instances of this option in a single <i>cc</i> command invocation. |  |
| 6527 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6528 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6529 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6530 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6531 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6532 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6533 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 6534 |       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |

6535        **-U *name***  
 6536            Remove any initial definition of *name*, where *name* is a reserved symbol that is  
 6537            predefined by the particular preprocessor.

6538        The following options are passed by *cc* (with their associated arguments) to the link-edit phase:

6539        **-e *epsym***  
 6540            Set the default entry point address for the output file to be that of the symbol *epsym*.

6541        **-L *dir***  
 6542            Change the algorithm of searching for the libraries named in the **-l** objects to look in  
 6543            the directory named by the *directory* pathname before looking in the usual places.  
 6544            Directories named in **-L** options will be searched in the order specified.  
 6545            Implementations support at least ten instances of this option in a single *cc* command  
 6546            invocation. If a directory specified by a **-L** option contains files named **libc.a**, **libm.a**,  
 6547            **libl.a** or **liby.a**, the results are unspecified. This option is only effective if it precedes  
 6548            the **-l** option on the command line.

6549        **-r**        Retain relocation entries in the output object file. Relocation entries must be saved if  
 6550            the output is to become the input of a subsequent *cc* run. The link-edit phase will not  
 6551            complain about unresolved references and will not make the object output executable.

6552        **-s**        Produce object or executable files, or both, from which symbolic and other information  
 6553            not required for proper execution using the **XSH** specification *exec* family has been  
 6554            removed (stripped). If both **-g** and **-s** options are present, the action taken is  
 6555            unspecified.

6556        **-u *symname***  
 6557            Enter *symname* as an undefined symbol into the symbol table. This is useful for loading  
 6558            entirely from a library, since initially the symbol table is empty and an unresolved  
 6559            reference is needed to force loading of the first routine.

## 6560 OPERANDS

6561        An *operand* is either in the form of a pathname or the form **-l *library***. At least one operand of the  
 6562        pathname form must be specified. The following operands are supported:

6563        ***file.c***     A C-language source file that may be preprocessed, compiled, optimised and link  
 6564            edited.

6565        ***file.i***     A C-language source file that has been preprocessed, and may be compiled, optimised  
 6566            and link edited.

6567        ***file.s***     An assembly language source file that may be assembled and link edited.

6568        ***file.a***     A library of object files typically produced by the *ar* utility, and passed directly to the  
 6569            link editor.

6570        The operand must be one of the forms *file.c*, *file.i* or *file.s* if the **-c** option is used.

6571        **-l *library***  
 6572            (The letter ell.) Search the library named:

6573                *liblibrary.a*

6574        A library will be searched when its name is encountered, so the placement of a **-l**  
 6575        operand is significant. Several standard libraries can be specified in this manner, as  
 6576        described in the EXTENDED DESCRIPTION section.

6577 Other arguments are taken to be C-language compatible object programs, typically produced by  
 6578 an earlier *cc* run, or perhaps libraries of C-language compatible routines, and are passed directly  
 6579 to the link editor. These programs, together with the results of any compilations specified, are  
 6580 linked (in the order given) to produce an executable program with the name **a.out** (unless the **-o**  
 6581 link-edit option is used).

6582 The standard C-language library is automatically available to the C-language program. Other  
 6583 libraries may be specified explicitly using the **-l** option with *cc*.

#### 6584 **STDIN**

6585 Not used.

#### 6586 **INPUT FILES**

6587 The input file will be one of the following: a text file containing a C-language source program; a  
 6588 text file containing an (implementation-dependent) assembly-language source program; an  
 6589 object file in the format produced by *cc -c* or a library of object files, in the format produced by  
 6590 archiving zero or more object files, using *ar*. Additional input file formats are implementation-  
 6591 dependent.

#### 6592 **ENVIRONMENT VARIABLES**

6593 The following environment variable affects the execution of *cc*:

##### 6594 ***TMPDIR***

6595 Provide a pathname that will override the default directory for temporary files, if any.

6596 The following environment variables may affect the execution of *cc*:

6597 ***LANG*** Provide a default value for the internationalisation variables that are unset or null. If  
 6598 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 6599 default locale will be used. If any of the internationalisation variables contains an  
 6600 invalid setting, the utility will behave as if none of the variables had been defined.

##### 6601 ***LC\_ALL***

6602 If set to a non-empty string value, override the values of all the other  
 6603 internationalisation variables.

##### 6604 ***LC\_CTYPE***

6605 Determine the locale for the interpretation of sequences of bytes of text data as  
 6606 characters (for example, single- as opposed to multi-byte characters in arguments and  
 6607 input files).

##### 6608 ***LC\_MESSAGES***

6609 Determine the locale that should be used to affect the format and contents of diagnostic  
 6610 messages written to standard error.

##### 6611 ***NLSPATH***

6612 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

#### 6613 **ASYNCHRONOUS EVENTS**

6614 Default.

#### 6615 **STDOUT**

6616 If more than one file operand ending in *.c*, *.i* or *.s* is given, for each such file:

6617 `"%s:\n", <file>`

6618 may be written. These messages, if written, will precede the processing of each input file; they  
 6619 will not be written to standard output if they are written to standard error, as described in the  
 6620 **STDERR** section.

6621 If the **-E** option is specified, the standard output will be a text file that represents the results of  
6622 the preprocessing stage of the language; it may contain extra information appropriate for  
6623 subsequent compilation passes.

#### 6624 **STDERR**

6625 Used only for diagnostic messages. If more than one file operand ending in **.c** (or possibly other  
6626 unspecified suffixes) is given, for each such file:

6627       "**%s:\n**", *<file>*

6628 may be written to allow identification of the diagnostic and warning messages with the  
6629 appropriate input file. These messages, if written, will precede the processing of each input file;  
6630 they will not be written to standard error if they are written to standard output, as described in  
6631 the **STDOUT** section.

6632 This utility may produce warning messages about certain conditions that do not warrant  
6633 returning an error (non-zero) exit value.

#### 6634 **OUTPUT FILES**

6635 If the **-P** option is specified, text files are created that represent the results of the preprocessing  
6636 stage of the language.

6637 Object files or executable files or both are produced in unspecified formats.

#### 6638 **EXTENDED DESCRIPTION**

6639 All implementations will support standard libraries as described in *c89*, **EXTENDED**  
6640 **DESCRIPTION**.

#### 6641 **External Symbols**

6642 The C compiler and link editor support the significance of external symbols up to a length of at  
6643 least 31 bytes; the action taken upon encountering symbols exceeding the implementation-  
6644 dependent maximum symbol length is unspecified.

6645 The compiler and link editor support a minimum of 511 external symbols per source or object  
6646 file, and a minimum of 4095 external symbols total. A diagnostic message will be written to the  
6647 standard output if the implementation-dependent limit is exceeded; other actions are  
6648 unspecified.

#### 6649 **Programming Environment**

6650 All implementations will support one or more programming environments with *cc* as specified  
6651 in *c89*, **EXTENDED DESCRIPTION**.

#### 6652 **EXIT STATUS**

6653 The following exit values are returned:

6654       **0** Successful compilation or link edit.

6655       **>0** An error occurred.

#### 6656 **CONSEQUENCES OF ERRORS**

6657 When *cc* encounters a compilation error that causes an object file not to be created, it will write a  
6658 diagnostic to standard error and continue to compile other source code operands, but it will not  
6659 perform the link phase and will return a non-zero exit status. If the link edit is unsuccessful, a  
6660 diagnostic message will be written to standard error and *cc* will exit with a non-zero status. A  
6661 portable application must rely on the exit status of *cc*, rather than on the existence or mode of the  
6662 executable file.

6663 **APPLICATION USAGE**

6664 The *c89* utility provides an interface to the ISO C standard, but the *cc* utility accepts an  
 6665 unspecified dialect of the C language: it may be Standard C, common-usage C or some other  
 6666 variant. Portable C programs should be written to conform to the ISO C standard and compiled  
 6667 with *c89*.

6668 Since the *cc* utility usually creates files in the current directory during the compilation process, it  
 6669 is typically necessary to run the *cc* utility in a directory in which a file can be created.

6670 Some historical implementations have created *.o* files when *-c* is not specified and more than  
 6671 one source file is given. Since this area is left unspecified, the application cannot rely on *.o* files  
 6672 being created, but it also must be prepared for any related *.o* files that already exist being deleted  
 6673 at the completion of the link edit.

6674 Some historical implementations have permitted *-L* options to be interspersed with *-l* operands  
 6675 on the command line. For an application to compile consistently on systems that do not behave  
 6676 like this, it is necessary for a portable application to supply all *-L* options before any of the *-l*  
 6677 options.

6678 There is the possible implication that if a user supplies versions of the standard library functions  
 6679 (before they would be encountered by an implicit *-l c* or explicit *-l m*), that those versions  
 6680 would be used in place of the standard versions. There are various reasons this might not be  
 6681 true (functions defined as macros, manipulations for clean name space, and so on), so the  
 6682 existence of files named in the same manner as the standard libraries within the *-L* directories is  
 6683 explicitly stated to produce unspecified behaviour.

6684 All of the interfaces specified in the **XSH** specification may be made visible by implementations  
 6685 when the Standard C Library is searched. Portable applications must explicitly request  
 6686 searching the other standard libraries when functions made visible by those libraries are used.

6687 Applications should migrate to the *c89* utility.

6688 **EXAMPLES**

6689 The following are examples of usage:

6690 `cc -o foo foo.c bar.s`

6691 Compiles **foo.c**, assembles **bar.s** and creates the executable file **foo**.

6692 `cc -c foo.c`

6693 Compiles **foo.c** and creates the object file **foo.o**.

6694 `cc foo.c`

6695 Compiles **foo.c** and creates the executable file **a.out**.

6696 `cc foo.c bar.o`

6697 Compiles **foo.c**, links it with **bar.o**, and creates the executable **a.out**. Also creates and  
 6698 leaves **foo.o**.

6699 The following examples clarify the use and interactions of *-L* options and *-l* operands:

- 6700 1. Consider the case in which module **a.c** calls function *f()* in library **libQ.a**, and module **b.c**  
 6701 calls function *g()* in library **libp.a**. Assume that both libraries reside in */a/b/c*. The  
 6702 command line to compile and link in the desired way is:

6703 `cc -L /a/b/c main.o a.c -l Q b.c -l p`

6704 In this case the *-l Q* operand need only precede the first *-l p* operand, since both **libQ.a**  
 6705 and **libp.a** reside in the same directory.

6706 2. Multiple **-L** operands can be used when library name collisions occur. Building on the  
 6707 previous example, suppose that the user now wants to use a new **libp.a**, in **/a/a/a**, but still  
 6708 wants **f()** from **/a/b/c/libQ.a**:

6709 `cc -L /a/a/a -L /a/b/c main.o a.c -l Q b.c -l p`

6710 In this example, the linker searches the **-L** options in the order specified, and finds  
 6711 **/a/a/a/libp.a** before **/a/b/c/libp.a** when resolving references for **b.c**. The order of the **-l**  
 6712 operands is still important, however.

#### 6713 FUTURE DIRECTIONS

6714 None.

#### 6715 SEE ALSO

6716 *ar, c89, nm, sdb, strip.*

#### 6717 CHANGE HISTORY

6718 First released in Issue 2.

#### 6719 Issue 4

6720 Format reorganised.

6721 Internationalised environment variable support made optional.

6722 Utility Syntax Guidelines support mandated.

#### 6723 Issue 4, Version 2

6724 In the **Standard Libraries** subsection, the **-l c** operand describes access to traditional interfaces  
 6725 if **\_XOPEN\_UNIX** is defined.

#### 6726 Issue 5

6727 The EXTENDED DESCRIPTION is changed to reference *c89* for standard libraries.

6728 Marked LEGACY.



6729 **NAME**6730 `cd` — change the working directory6731 **SYNOPSIS**6732 `cd` [*directory*]6733 EX `cd -`6734 **DESCRIPTION**

6735 The `cd` utility will change the working directory of the current shell execution environment; see  
 6736 EX Section 2.12 on page 63. If the current working directory is successfully changed, it will save an  
 6737 absolute pathname of the old working directory in the environment variable `OLDPWD` and it  
 6738 will save an absolute pathname of the new working directory in the environment variable `PWD`.

6739 When invoked with no operands, and the `HOME` environment variable is set to a non-empty  
 6740 value, the directory named in the `HOME` environment variable will become the new working  
 6741 directory. If `HOME` is empty or is undefined, the default behaviour is implementation-  
 6742 dependent.

6743 **OPTIONS**

6744 None.

6745 **OPERANDS**

6746 The following operands are supported:

6747 *directory*

6748 An absolute or relative pathname of the directory that becomes the new working  
 6749 directory. The interpretation of a relative pathname by `cd` depends on the `CDPATH`  
 6750 environment variable.

6751 EX `-` When a hyphen is used as the operand, this is equivalent to the command:

6752 `cd "$OLDPWD" && pwd`

6753 which changes to the previous working directory and then writes its name.

6754 **STDIN**

6755 Not used.

6756 **INPUT FILES**

6757 None.

6758 **ENVIRONMENT VARIABLES**6759 The following environment variables affect the execution of `cd`:6760 `CDPATH`

6761 A colon-separated list of pathnames that refer to directories. If the *directory* operand  
 6762 does not begin with a slash (/) character, and the first component is not dot or dot-dot,  
 6763 `cd` will search for *directory* relative to each directory named in the `CDPATH` variable, in  
 6764 the order listed. The new working directory will be set to the first matching directory  
 6765 found. An empty string in place of a directory pathname represents the current  
 6766 directory. If `CDPATH` is not set, it will be treated as if it were an empty string.

6767 `HOME` The name of the home directory, used when no *directory* operand is specified.

6768 `LANG` Provide a default value for the internationalisation variables that are unset or null. If  
 6769 `LANG` is unset or null, the corresponding value from the implementation-dependent  
 6770 default locale will be used. If any of the internationalisation variables contains an  
 6771 invalid setting, the utility will behave as if none of the variables had been defined.

6772 **LC\_ALL**  
 6773 If set to a non-empty string value, override the values of all the other  
 6774 internationalisation variables.

6775 **LC\_CTYPE**  
 6776 Determine the locale for the interpretation of sequences of bytes of text data as  
 6777 characters (for example, single- as opposed to multi-byte characters in arguments).

6778 **LC\_MESSAGES**  
 6779 Determine the locale that should be used to affect the format and contents of diagnostic  
 6780 messages written to standard error.

6781 EX **NLSPATH**  
 6782 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

6783 EX **OLDPWD**  
 6784 A pathname of the previous working directory, used by *cd -*.

6785 EX **PWD** A pathname of the current working directory, set by *cd* after it has changed to that  
 6786 directory.

6787 **ASYNCHRONOUS EVENTS**  
 6788 Default.

6789 **STDOUT**  
 6790 EX If a non-empty directory name from *CDPATH* is used, or if *cd -* is used, an absolute pathname of  
 6791 the new working directory will be written to the standard output as follows:  
 6792 "%s\n", <new directory>  
 6793 Otherwise, there will be no output.

6794 **STDERR**  
 6795 Used only for diagnostic messages.

6796 **OUTPUT FILES**  
 6797 None.

6798 **EXTENDED DESCRIPTION**  
 6799 None.

6800 **EXIT STATUS**  
 6801 The following exit values are returned:  
 6802 0 The directory was successfully changed.  
 6803 >0 An error occurred.

6804 **CONSEQUENCES OF ERRORS**  
 6805 The working directory remains unchanged.

6806 **APPLICATION USAGE**  
 6807 Since *cd* affects the current shell execution environment, it is always provided as a shell regular  
 6808 built-in. If it is called in a subshell or separate utility execution environment, such as one of the  
 6809 following:  
 6810 (cd /tmp)  
 6811 nohup cd  
 6812 find . -exec cd {} \;  
 6813 it will not affect the working directory of the caller's environment.

6814           The user must have execute (search) permission in *directory* in order to change to it.

6815 **EXAMPLES**

6816           None.

6817 **FUTURE DIRECTIONS**

6818           None.

6819 **SEE ALSO**

6820           *pwd*, the **XSH** specification description of *chdir()*.

6821 **CHANGE HISTORY**

6822           First released in Issue 2.

6823 **Issue 4**

6824           Aligned with the ISO/IEC 9945-2: 1993 standard.

6825           Extensions added for *cd-*, *PWD* and *OLDPWD*.

## 6826 NAME

6827 cflow — generate a C-language flowgraph (DEVELOPMENT)

## 6828 SYNOPSIS

```
6829 EX cflow [r][-d num][-D name[=def]] ... [-i incl][-I dir] ... [-U dir] ...
6830 file ...
```

## 6831 DESCRIPTION

6832 The *cflow* utility analyses a collection of object files or assembler, C-language, *lex* or *yacc* source  
 6833 files, and attempts to build a graph, written to standard output, charting the external references.

## 6834 OPTIONS

6835 The *cflow* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 6836 that the order of the **-D**, **-I** and **-U** options (which are identical to their interpretation by *c89*) is  
 6837 significant. The following options are supported:

6838 **-d num**

6839 Indicate the depth at which the flowgraph is cut off. The argument *num* is a decimal  
 6840 integer. By default this is a very large number (typically greater than 32 000). Attempts  
 6841 to set the cut-off depth to a non-positive integer will be ignored.

6842 **-i incl** Increase the number of included symbols. The *incl* option-argument is one of the  
 6843 following characters:

6844 **x** Include external and static data symbols. The default is to include only functions  
 6845 in the flowgraph.

6846 **\_** (Underscore) Include names that begin with an underscore. The default is to  
 6847 exclude these functions (and data if **-i x** is used).

6848 **-r** Reverse the caller: callee relationship, producing an inverted listing showing the callers  
 6849 of each function. The listing is also sorted in lexicographical order by callee.

## 6850 OPERANDS

6851 The following operand is supported:

6852 **file** The pathname of a file for which a graph is to be generated. Files suffixed in *.l*, *.y*, *.c*  
 6853 and *.i* are processed by *lex* and *yacc* and preprocessed by the *c89* preprocessor phase  
 6854 (bypassed for *.i* files) as appropriate, and then run through the first pass of *lint*. Files  
 6855 suffixed with *.s* are assembled and information is extracted (as in *.o* files) from the  
 6856 symbol table.

## 6857 STDIN

6858 Not used.

## 6859 INPUT FILES

6860 The input files are object files or assembler, C-language, *lex* or *yacc* source files.

## 6861 ENVIRONMENT VARIABLES

6862 The following environment variables affect the execution of *cflow*:

6863 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 6864 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 6865 default locale will be used. If any of the internationalisation variables contains an  
 6866 invalid setting, the utility will behave as if none of the variables had been defined.

6867 **LC\_ALL**

6868 If set to a non-empty string value, override the values of all the other  
 6869 internationalisation variables.

6870 **LC\_COLLATE**  
 6871 Determine the locale for the ordering of the output when the **-r** option is used.

6872 **LC\_CTYPE**  
 6873 Determine the locale for the interpretation of sequences of bytes of text data as  
 6874 characters (for example, single- as opposed to multi-byte characters in arguments and  
 6875 input files).

6876 **LC\_MESSAGES**  
 6877 Determine the locale that should be used to affect the format and contents of diagnostic  
 6878 messages written to standard error.

6879 **NLSPATH**  
 6880 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

6881 **ASYNCHRONOUS EVENTS**  
 6882 Default.

6883 **STDOUT**  
 6884 The flowgraph written to standard output is formatted as follows:  
 6885 `"%d %s:%s\n", <reference number>, <global>, <definition>`  
 6886 Each line of output begins with a reference (that is, line) number, followed by a suitable amount  
 6887 of indentation indicating the level. This is followed by the name of the global, a colon and its  
 6888 definition. Normally globals are only functions not defined as an external or beginning with an  
 6889 underscore; see the **OPTIONS** section for the **-i** inclusion option. For information extracted from  
 6890 C-language source, the definition consists of an abstract type declaration (for example, **char \***)  
 6891 and, delimited by angle brackets, the name of the source file and the line number where the  
 6892 definition was found. Definitions extracted from object files indicate the filename and location  
 6893 counter under which the symbol appeared (for example, *text*).  
 6894 Once a definition of a name has been written, subsequent references to that name contain only  
 6895 the reference number of the line where the definition can be found. For undefined references,  
 6896 only **< >** is written.

6897 **STDERR**  
 6898 Used only for diagnostic messages.

6899 **OUTPUT FILES**  
 6900 None.

6901 **EXTENDED DESCRIPTION**  
 6902 None.

6903 **EXIT STATUS**  
 6904 The following exit values are returned:  
 6905 **0** Successful completion.  
 6906 **>0** An error occurred.

6907 **CONSEQUENCES OF ERRORS**  
 6908 Default.

6909 **APPLICATION USAGE**  
 6910 Files produced by *lex* and *yacc* cause the reordering of line number declarations, and this can  
 6911 confuse *cflow*. To obtain proper results, the input of *yacc* or *lex* must be directed to *cflow*.

6912 **EXAMPLES**6913       Given the following in **file.c**:

```
6914 int i;
6915 main()
6916 {
6917 f();
6918 g();
6919 f();
6920 }
6921 f()
6922 {
6923 i = h();
6924 }
```

6925       The command:

6926       cflow -i x file.c

6927       produces the output:

```
6928 1 main: int(), <file.c 4>
6929 2 f: int(), <file.c 11>
6930 3 h: <>
6931 4 i: int, <file.c 1>
6932 5 g: <>
```

6933 **FUTURE DIRECTIONS**

6934       None.

6935 **SEE ALSO**

6936       cc, c89, lex, yacc.

6937 **CHANGE HISTORY**

6938       First released in Issue 2.

6939 **Issue 4**

6940       Format reorganised.

6941       Internationalised environment variable support mandated.

6942 **NAME**

6943       chgrp — change the file group ownership

6944 **SYNOPSIS**6945       chgrp [-R] *group file ...*6946 **DESCRIPTION**6947       The *chgrp* utility will set the group ID of the file named by each *file* operand to the group ID specified by the *group* operand.6949       For each *file* operand, it will perform actions equivalent to the **XSH** specification *chown()* function, called with the following arguments:

- 6951       • The *file* operand will be used as the *path* argument.
- 6952       • The user ID of the file will be used as the *owner* argument.
- 6953       • The specified group ID will be used as the *group* argument.

6954       Unless *chgrp* is invoked by a process with appropriate privileges, the set-user-ID and set-group-ID bits of a regular file will be cleared upon successful completion; the set-user-ID and set-group-ID bits of other file types may be cleared.

6957 **OPTIONS**6958       The *chgrp* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

6959       The following option is supported:

- 6960       **-R**       Recursively change file group IDs. For each *file* operand that names a directory, *chgrp* will change the group of the directory and all files in the file hierarchy below it.

6962 **OPERANDS**

6963       The following operands are supported:

- 6964       *group*     A group name from the group database or a numeric group ID. Either specifies a group ID to be given to each file named by one of the *file* operands. If a numeric *group* operand exists in the group database as a group name, the group ID number associated with that group name is used as the group ID.
- 6968       *file*     A pathname of a file whose group ID is to be modified.

6969 **STDIN**

6970       Not used.

6971 **INPUT FILES**

6972       None.

6973 **ENVIRONMENT VARIABLES**6974       The following environment variables affect the execution of *chgrp*:

- 6975       **LANG**     Provide a default value for the internationalisation variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

6979       **LC\_ALL**

6980       If set to a non-empty string value, override the values of all the other internationalisation variables.

6982       **LC\_CTYPE**

6983       Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments).

6985 *LC\_MESSAGES*  
 6986 Determine the locale that should be used to affect the format and contents of diagnostic  
 6987 messages written to standard error.

6988 EX *NLSPATH*  
 6989 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

6990 **ASYNCHRONOUS EVENTS**  
 6991 Default.

6992 **STDOUT**  
 6993 Not used.

6994 **STDERR**  
 6995 Used only for diagnostic messages.

6996 **OUTPUT FILES**  
 6997 None.

6998 **EXTENDED DESCRIPTION**  
 6999 None.

7000 **EXIT STATUS**  
 7001 The following exit values are returned:  
 7002 0 The utility executed successfully and all requested changes were made.  
 7003 >0 An error occurred.

7004 **CONSEQUENCES OF ERRORS**  
 7005 If, when invoked with the **-R** option, *chgrp* attempts but fails to change the group ID of a  
 7006 particular file in a specified file hierarchy, it will continue to process the remaining files in the  
 7007 hierarchy. If *chgrp* cannot read or search a directory within a hierarchy, it will continue to  
 7008 process the other parts of the hierarchy that are accessible.

7009 **APPLICATION USAGE**  
 7010 Only the owner of a file or the user with appropriate privileges may change the owner or group  
 7011 of a file.  
 7012 Some systems restrict the use of *chgrp* to a user with appropriate privileges when the *group*  
 7013 specified is not the effective group ID or one of the supplementary group IDs of the calling  
 7014 process.

7015 **EXAMPLES**  
 7016 None.

7017 **FUTURE DIRECTIONS**  
 7018 None.

7019 **SEE ALSO**  
 7020 *chmod*, *chown*, the **XSH** specification description of *chown*().

7021 **CHANGE HISTORY**  
 7022 First released in Issue 2.

7023 **Issue 4**  
 7024 Aligned with the ISO/IEC 9945-2: 1993 standard.



## 7025 NAME

7026 chmod — change the file modes

## 7027 SYNOPSIS

7028 chmod [-R] *mode file...*

## 7029 DESCRIPTION

7030 The *chmod* utility will change any or all of the file mode bits of the file named by each *file*  
7031 operand in the way specified by the *mode* operand.7032 It is implementation-dependent whether and how the *chmod* utility affects any alternate or  
7033 additional file access control mechanism (see **file access permissions** in the **XBD** specification,  
7034 **Chapter 2, Glossary**) being used for the specified file.7035 Only a process whose effective user ID matches the user ID of the file, or a process with the  
7036 appropriate privileges, will be permitted to change the file mode bits of a file.

## 7037 OPTIONS

7038 The *chmod* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

7039 The following option is supported:

7040 **-R** Recursively change file mode bits. For each *file* operand that names a directory, *chmod*  
7041 will change the file mode bits of the directory and all files in the file hierarchy below it.

## 7042 OPERANDS

7043 The following operands are supported:

7044 *mode* Represents the change to be made to the file mode bits of each file named by one of the  
7045 *file* operands; see the EXTENDED DESCRIPTION section.7046 *file* A pathname of a file whose file mode bits are to be modified.

## 7047 STDIN

7048 Not used.

## 7049 INPUT FILES

7050 None.

## 7051 ENVIRONMENT VARIABLES

7052 The following environment variables affect the execution of *chmod*:7053 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
7054 *LANG* is unset or null, the corresponding value from the implementation-dependent  
7055 default locale will be used. If any of the internationalisation variables contains an  
7056 invalid setting, the utility will behave as if none of the variables had been defined.7057 *LC\_ALL*7058 If set to a non-empty string value, override the values of all the other  
7059 internationalisation variables.7060 *LC\_CTYPE*7061 Determine the locale for the interpretation of sequences of bytes of text data as  
7062 characters (for example, single- as opposed to multi-byte characters in arguments).7063 *LC\_MESSAGES*7064 Determine the locale that should be used to affect the format and contents of diagnostic  
7065 messages written to standard error.7066 EX *NLSPATH*7067 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

7068 **ASYNCHRONOUS EVENTS**

7069       Default.

7070 **STDOUT**

7071       Not used.

7072 **STDERR**

7073       Used only for diagnostic messages.

7074 **OUTPUT FILES**

7075       None.

7076 **EXTENDED DESCRIPTION**

7077 **EX**       The *mode* operand will be either a **symbolic\_mode** expression or a non-negative octal integer.  
 7078       The **symbolic\_mode** form is described by the grammar later in this section.

7079       Each **clause** will specify an operation to be performed on the current file mode bits of each *file*.  
 7080       The operations will be performed on each *file* in the order in which the **clauses** are specified.

7081       The *who* symbols u, g and o will specify the *user*, *group* and *other* parts of the file mode bits,  
 7082       respectively. A *who* consisting of the symbol a will be equivalent to **ugo**.

7083       The *perm* symbols r, w and x represent the *read*, *write* and *execute/search* portions of file mode bits,  
 7084       respectively. The *perm* symbol s represent the *set-user-ID-on-execution* (when **who** contains or  
 7085       implies u) and *set-group-ID-on-execution* (when **who** contains or implies g) bits.

7086       The **perm** symbol X represent the execute/search portion of the file mode bits if the file is a  
 7087       directory or if the current (unmodified) file mode bits have at least one of the execute bits  
 7088       (S\_IXUSR, S\_IXGRP or S\_IXOTH) set. It will be ignored if the file is not a directory and none of  
 7089       the execute bits are set in the current file mode bits.

7090       The **permcoppy** symbols u, g and o represent the current permissions associated with the user,  
 7091       group and other parts of the file mode bits, respectively. For the remainder of this section, **perm**  
 7092       refers to the non-terminals **perm** and **permcoppy** in the grammar.

7093       If multiple **actionlists** are grouped with a single **wholist** in the grammar, each **actionlist** will be  
 7094       applied in the order specified with that **wholist**. The **op** symbols represent the operation  
 7095       performed, as follows:

- 7096       +    If **perm** is not specified, the + operation will not change the file mode bits.  
 7097            If **who** is not specified, the file mode bits represented by **perm** for the owner, group and  
 7098            other permissions, except for those with corresponding bits in the file mode creation mask  
 7099            of the invoking process, will be set.  
 7100            Otherwise, the file mode bits represented by the specified **who** and **perm** values will be set.
- 7101       –    If **perm** is not specified, the "-" operation will not change the file mode bits. |  
 7102            If **who** is not specified, the file mode bits represented by **perm** for the owner, group and  
 7103            other permissions, except for those with corresponding bits in the file mode creation mask  
 7104            of the invoking process, will be cleared.
- 7105            Otherwise, the file mode bits represented by the specified **who** and **perm** values will be  
 7106            cleared.
- 7107       =    Clear the file mode bits specified by the **who** value, or, if no **who** value is specified, all of the  
 7108            file mode bits specified in this specification. |
- 7109       If **perm** is not specified, the = operation will make no further modifications to the file mode  
 7110       bits.

If **who** is not specified, the file mode bits represented by **perm** for the owner, group and other permissions, except for those with corresponding bits in the file mode creation mask of the invoking process, will be set.

Otherwise, the file mode bits represented by the specified **who** and **perm** values will be set.

When using the symbolic mode form on a regular file, it is implementation-dependent whether or not:

- Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are ignored.
- Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-on-execution bits.
- Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all execute bits are currently clear are ignored. However, if the command `ls -l file` writes an `s` in the position indicating that the set-user-ID-on-execution or set-group-ID-on-execution is set, the commands `chmod u-s file` or `chmod g-s file`, respectively, will not be ignored.

When using the symbolic mode form on other file types, it is implementation-dependent whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honoured.

If the **who** symbol `o` is used in conjunction with the **perm** symbol `s` with no other **who** symbols being specified, the set-user-ID-on-execution and set-group-ID-on-execution bits will not be modified. It will not be an error to specify the **who** symbol `o` in conjunction with the **perm** symbol `s`.

For an octal integer *mode* operand, the file mode bits will be set absolutely.

For each bit set in the octal number, the corresponding file permission bit shown in the following table will be set; all other file permission bits will be cleared. For regular files, for each bit set in the octal number corresponding to the set-user-ID-on-execution or the set-group-ID-on-execution bits shown in the following table will be set; if these bits are not set in the octal number, they will be cleared. For other file types, it is implementation-dependent whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honoured.

| Octal | Mode bit | Octal | Mode bit | Octal | Mode bit | Octal | Mode bit |
|-------|----------|-------|----------|-------|----------|-------|----------|
| 4000  | S_ISUID  | 0400  | S_IRUSR  | 0040  | S_IRGRP  | 0004  | S_IROTH  |
| 2000  | S_ISGID  | 0200  | S_IWUSR  | 0020  | S_IWGRP  | 0002  | S_IWOTH  |
|       |          | 0100  | S_IXUSR  | 0010  | S_IXGRP  | 0001  | S_IXOTH  |

When bits are set in the octal number other than those listed in the table above, the behaviour is unspecified.

### Grammar for chmod

The grammar and lexical conventions in this section describe the syntax for the **symbolic\_mode** operand. The general conventions for this style of grammar are described in Section 1.8 on page 10. A valid **symbolic\_mode** can be represented as the non-terminal symbol **symbolic\_mode** in the grammar. This formal syntax takes precedence over the preceding text syntax description.

The lexical processing will be based entirely on single characters. Implementations need not allow blank characters within the single argument being processed.

```

7154 %start symbolic_mode
7155 %%
7156 symbolic_mode : section
7157 | symbolic_mode ',' section
7158 ;
7159 section : actionlist
7160 | wholist actionlist
7161 ;
7162 wholist : who
7163 | wholist who
7164 ;
7165 who : 'u' | 'g' | 'o' | 'a'
7166 ;
7167 actionlist : action
7168 | actionlist action
7169 ;
7170 action : op
7171 | op permlist
7172 | op permcopy
7173 ;
7174 permcopy : 'u' | 'g' | 'o'
7175 ;
7176 op : '+' | '-' | '='
7177 ;
7178 permlist : perm
7179 | perm permlist
7180 ;
7181 perm : 'r' | 'w' | 'x' | 'X' | 's'
7182 ;

```

### 7183 EXIT STATUS

7184 The following exit values are returned:

7185     0   The utility executed successfully and all requested changes were made.  
7186     >0   An error occurred.

### 7187 CONSEQUENCES OF ERRORS

7188 If, when invoked with the **-R** option, *chmod* attempts but fails to change the mode of a particular  
7189 file in a specified file hierarchy, it will continue to process the remaining files in the hierarchy,  
7190 affecting the final exit status. If *chmod* cannot read or search a directory within a hierarchy, it  
7191 will continue to process the other parts of the hierarchy that are accessible.

### 7192 APPLICATION USAGE

7193 The references to octal modes are marked **EX** because, although they are obsolescent in the  
7194 ISO/IEC 9945-2:1993 standard, XSI-conformant systems have committed to maintaining them  
7195 for portable applications until further notice.

7196 Some implementations of the *chmod* utility change the mode of a directory before the files in the  
7197 directory when performing a recursive (**-R** option) change; others change the directory mode  
7198 after the files in the directory. If an application tries to remove read or search permission for a

7199 file hierarchy, the removal attempt will fail if the directory is changed first; on the other hand,  
 7200 trying to re-enable permissions to a restricted hierarchy will fail if directories are changed last.  
 7201 Users should not try to make a hierarchy inaccessible to themselves.

7202 Some implementations of *chmod* never used the process' *umask* when changing modes; systems  
 7203 conformant with this specification do so when **who** is not specified. Note the difference  
 7204 between:

7205 `chmod a-w file`

7206 which removes all write permissions, and:

7207 `chmod -- -w file`

7208 which removes write permissions that would be allowed if **file** was created with the same  
 7209 *umask*.

7210 Portable applications should never assume that they know how the set-user-ID and set-group-  
 7211 ID bits on directories will be interpreted.

## 7212 EXAMPLES

| 7213 | Mode         | Results                                                                                                            |
|------|--------------|--------------------------------------------------------------------------------------------------------------------|
| 7214 |              |                                                                                                                    |
| 7215 | <b>a+=</b>   | Equivalent to <code>a+ , a=</code> ; clears all file mode bits.                                                    |
| 7216 | <b>go+-w</b> | Equivalent to <code>go+ , go-w</code> ; clears group and other write bits.                                         |
| 7217 | <b>g=o-w</b> | Equivalent to <code>g=o , g-w</code> ; sets group bit to match other bits and then clears group<br>7218 write bit. |
| 7219 | <b>g-r+w</b> | Equivalent to <code>g-r , g+w</code> ; clears group read bit and sets group write bit.                             |
| 7220 | <b>=g</b>    | Sets owner bits to match group bits and sets other bits to match group bits.                                       |

## 7221 FUTURE DIRECTIONS

7222 None.

## 7223 SEE ALSO

7224 *ls*, *umask*, the XSH specification description of *chmod*(.).

## 7225 CHANGE HISTORY

7226 First released in Issue 2.

## 7227 Issue 4

7228 Aligned with the ISO/IEC 9945-2: 1993 standard.

7229 **NAME**

7230 chown — change the file ownership

7231 **SYNOPSIS**

7232 chown [-R] owner[:group] file ...

7233 **DESCRIPTION**7234 The *chown* utility will set the user ID of the file named by each *file* operand to the user ID  
7235 specified by the *owner* operand.7236 For each *file* operand, it will perform actions equivalent to the **XSH** specification *chown()*  
7237 function, called with the following arguments:

- 7238 1. The *file* operand will be used as the *path* argument.
- 7239 2. The user ID indicated by the *owner* portion of the first operand will be used as the *owner*  
7240 argument.
- 7241 3. If the *group* portion of the first operand is given, the group ID indicated by it will be used  
7242 as the *group* argument; otherwise, the group ID of the file will be used as the *group*  
7243 argument.

7244 Unless *chown* is invoked by a process with appropriate privileges, the set-user-ID and set-  
7245 group-ID bits of a regular file will be cleared upon successful completion; the set-user-ID and  
7246 set-group-ID bits of other file types may be cleared.

7247 **OPTIONS**7248 The *chown* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

7249 The following option is supported:

- 7250 **-R** Recursively change file user IDs, and if the *group* operand is specified, group IDs. For  
7251 each *file* operand that names a directory, *chown* changes the user and group ID of the  
7252 directory and all files in the file hierarchy below it.

7253 **OPERANDS**

7254 The following operands are supported:

7255 *owner[:group]*

7256 A user ID and optional group ID to be assigned to *file*. The *owner* portion of this  
7257 operand must be a user name from the user database or a numeric user ID. Either  
7258 specifies a user ID to be given to each file named by one of the *file* operands. If a  
7259 numeric *owner* operand exists in the user database as a user name, the user ID number  
7260 associated with that user name will be used as the user ID. Similarly, if the *group*  
7261 portion of this operand is present, it must be a group name from the group database or  
7262 a numeric group ID. Either specifies a group ID to be given to each file. If a numeric  
7263 group operand exists in the group database as a group name, the group ID number  
7264 associated with that group name will be used as the group ID.

7265 *file* A pathname of a file whose user ID is to be modified.

7266 **STDIN**

7267 Not used.

7268 **INPUT FILES**

7269 None.

7270 **ENVIRONMENT VARIABLES**7271 The following environment variables affect the execution of *chown*:

7272 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 7273 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 7274 default locale will be used. If any of the internationalisation variables contains an  
 7275 invalid setting, the utility will behave as if none of the variables had been defined.

7276 **LC\_ALL**

7277 If set to a non-empty string value, override the values of all the other  
 7278 internationalisation variables.

7279 **LC\_CTYPE**

7280 Determine the locale for the interpretation of sequences of bytes of text data as  
 7281 characters (for example, single- as opposed to multi-byte characters in arguments).

7282 **LC\_MESSAGES**

7283 Determine the locale that should be used to affect the format and contents of diagnostic  
 7284 messages written to standard error.

7285 **EX NLSPATH**7286 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.7287 **ASYNCHRONOUS EVENTS**

7288 Default.

7289 **STDOUT**

7290 Not used.

7291 **STDERR**

7292 Used only for diagnostic messages.

7293 **OUTPUT FILES**

7294 None.

7295 **EXTENDED DESCRIPTION**

7296 None.

7297 **EXIT STATUS**

7298 The following exit values are returned:

7299 0 The utility executed successfully and all requested changes were made.

7300 &gt;0 An error occurred.

7301 **CONSEQUENCES OF ERRORS**

7302 If, when invoked with the **-R** option, *chown* attempts but fails to change the user ID or, if the  
 7303 *group* operand is specified, group ID, of a particular file in a specified file hierarchy, it will  
 7304 continue to process the remaining files in the hierarchy.

7305 If *chown* cannot read or search a directory within a hierarchy, it will continue to process the other  
 7306 parts of the hierarchy that are accessible.

7307 **APPLICATION USAGE**

7308 Only the owner of a file or the user with appropriate privileges may change the owner or group  
 7309 of a file.

7310 Some systems restrict the use of *chown* to a user with appropriate privileges.7311 **EXAMPLES**

7312 None.

7313 **FUTURE DIRECTIONS**

7314       None.

7315 **SEE ALSO**7316       *chmod*, *chgrp*, the **XSH** specification description of *chown*().7317 **CHANGE HISTORY**

7318       First released in Issue 2.

7319 **Issue 4**

7320       Aligned with the ISO/IEC 9945-2: 1993 standard.



7321 **NAME**

7322 cksum — write file checksums and sizes

7323 **SYNOPSIS**7324 cksum [*file* ...]7325 **DESCRIPTION**

7326 The *cksum* utility calculates and writes to standard output a cyclic redundancy check (CRC) for  
 7327 each input file, and also writes to standard output the number of octets in each file. The CRC  
 7328 used is based on the polynomial used for CRC error checking in the referenced Ethernet  
 7329 standard.

7330 The encoding for the CRC checksum is defined by the generating polynomial:

$$7331 \quad G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

7332 Mathematically, the CRC value corresponding to a given file is defined by the following  
 7333 procedure:

- 7334 1. The *n* bits to be evaluated are considered to be the coefficients of a mod 2 polynomial *M*(*x*)  
 7335 of degree *n*−1. These *n* bits are the bits from the file, with the most significant bit being the  
 7336 most significant bit of the first octet of the file and the last bit being the least significant bit  
 7337 of the last octet, padded with zero bits (if necessary) to achieve an integral number of  
 7338 octets, followed by one or more octets representing the length of the file as a binary value,  
 7339 least significant octet first. The smallest number of octets capable of representing this  
 7340 integer is used.
- 7341 2. *M*(*x*) is multiplied by  $x^{32}$  (that is, shifted left 32 bits) and divided by *G*(*x*) using mod 2  
 7342 division, producing a remainder *R*(*x*) of degree ≤ 31.
- 7343 3. The coefficients of *R*(*x*) are considered to be a 32-bit sequence.
- 7344 4. The bit sequence is complemented and the result is the CRC.

7345 **OPTIONS**

7346 None.

7347 **OPERANDS**

7348 The following operand is supported:

7349 *file* A pathname of a file to be checked. If no *file* operands are specified, the standard input  
 7350 is used.

7351 **STDIN**7352 The standard input is used only if no *file* operands are specified. See the INPUT FILES section.7353 **INPUT FILES**

7354 The input files can be any file type.

7355 **ENVIRONMENT VARIABLES**7356 The following environment variables affect the execution of *cksum*:

7357 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 7358 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 7359 default locale will be used. If any of the internationalisation variables contains an  
 7360 invalid setting, the utility will behave as if none of the variables had been defined.

7361 *LC\_ALL*

7362 If set to a non-empty string value, override the values of all the other  
 7363 internationalisation variables.

7364 *LC\_CTYPE*  
 7365 Determine the locale for the interpretation of sequences of bytes of text data as  
 7366 characters (for example, single- as opposed to multi-byte characters in arguments).

7367 *LC\_MESSAGES*  
 7368 Determine the locale that should be used to affect the format and contents of diagnostic  
 7369 messages written to standard error.

7370 EX *NLSPATH*  
 7371 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

7372 **ASYNCHRONOUS EVENTS**  
 7373 Default.

7374 **STDOUT**  
 7375 For each file processed successfully, the *cksum* utility will write in the following format:  
 7376 "%u %d %s\n", <checksum>, <# of octets>, <pathname>  
 7377 If no *file* operand was specified, the pathname and its leading space will be omitted.

7378 **STDERR**  
 7379 Used only for diagnostic messages.

7380 **OUTPUT FILES**  
 7381 None.

7382 **EXTENDED DESCRIPTION**  
 7383 None.

7384 **EXIT STATUS**  
 7385 The following exit values are returned:  
 7386 0 All files were processed successfully.  
 7387 >0 An error occurred.

7388 **CONSEQUENCES OF ERRORS**  
 7389 Default.

7390 **APPLICATION USAGE**  
 7391 The *cksum* utility is typically used to quickly compare a suspect file against a trusted version of  
 7392 the same, such as to ensure that files transmitted over noisy media arrive intact. However, this  
 7393 comparison cannot be considered cryptographically secure. The chances of a damaged file  
 7394 producing the same CRC as the original are small; deliberate deception is difficult, but probably  
 7395 not impossible.

7396 Although input files to *cksum* can be any type, the results need not be what would be expected  
 7397 on character special device files or on file types not described by the **XSH** specification. Since  
 7398 this specification does not specify the block size used when doing input, checksums of character  
 7399 special files need not process all of the data in those files.

7400 The algorithm is expressed in terms of a bitstream divided into octets. If a file is transmitted  
 7401 between two systems and undergoes any data transformation (such as moving 8-bit characters  
 7402 into 9-bit bytes or changing "Little Endian" byte ordering to "Big Endian"), identical CRC values  
 7403 cannot be expected. Implementations performing such transformations may extend *cksum* to  
 7404 handle such situations.

7405 **EXAMPLES**  
 7406 None.

7407 **FUTURE DIRECTIONS**

7408       None.

7409 **SEE ALSO**

7410       None.

7411 **CHANGE HISTORY**

7412       First released in Issue 4.

## 7413 NAME

7414 cmp — compare two files

## 7415 SYNOPSIS

7416 cmp [ -l | -s ] *file1 file2*

## 7417 DESCRIPTION

7418 The *cmp* utility compares two files. The *cmp* utility will write no output if the files are the same.  
 7419 Under default options, if they differ, it will write to standard output the byte and line number at  
 7420 which the first difference occurred. Bytes and lines will be numbered beginning with 1.

## 7421 OPTIONS

7422 The *cmp* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

7423 The following options are supported:

7424 **-l** (Lower-case ell.) Write the byte number (decimal) and the differing bytes (octal) for  
 7425 each difference.

7426 **-s** Write nothing for differing files; return exit status only.

## 7427 OPERANDS

7428 The following operands are supported:

7429 *file1* A pathname of the first file to be compared. If *file1* is "-", the standard input will be  
 7430 used.

7431 *file2* A pathname of the second file to be compared. If *file2* is "-", the standard input will be  
 7432 used.

7433 If both *file1* and *file2* refer to standard input or refer to the same FIFO special, block special or  
 7434 character special file, the results are undefined.

## 7435 STDIN

7436 The standard input will be used only if the *file1* or *file2* operand refers to standard input. See the  
 7437 INPUT FILES section.

## 7438 INPUT FILES

7439 The input files can be any file type.

## 7440 ENVIRONMENT VARIABLES

7441 The following environment variables affect the execution of *cmp*:

7442 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 7443 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 7444 default locale will be used. If any of the internationalisation variables contains an  
 7445 invalid setting, the utility will behave as if none of the variables had been defined.

7446 **LC\_ALL**

7447 If set to a non-empty string value, override the values of all the other  
 7448 internationalisation variables.

7449 **LC\_CTYPE**

7450 Determine the locale for the interpretation of sequences of bytes of text data as  
 7451 characters (for example, single- as opposed to multi-byte characters in arguments).

7452 **LC\_MESSAGES**

7453 Determine the locale that should be used to affect the format and contents of diagnostic  
 7454 messages written to standard error and informative messages written to standard  
 7455 output.

7456 EX **NLSPATH**  
 7457 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

7458 **ASYNCHRONOUS EVENTS**  
 7459 Default.

7460 **STDOUT**  
 7461 In the POSIX locale, results of the comparison will be written to standard output. When no  
 7462 options are used, the format will be:

7463 " %s %s differ: char %d, line %d\n", *file1*, *file2*, <*byte number*>,  
 7464 <*line number*>

7465 When the **-l** option is used, the format is:

7466 " %d %o %o\n", <*byte number*>, <*differing byte*>, <*differing byte*>

7467 for each byte that differs. The first <*differing byte*> number is from *file1* while the second is from  
 7468 *file2*. In both cases, <*byte number*> is relative to the beginning of the file, beginning with 1.

7469 No output will be written to standard output when the **-s** option is used.

7470 **STDERR**  
 7471 Used only for diagnostic messages. If *file1* and *file2* are identical for the entire length of the  
 7472 shorter file, in the POSIX locale the following diagnostic message will be written, unless the **-s**  
 7473 option is specified:

7474 "cmp: EOF on %s%s\n", <*name of shorter file*>, <*additional info*>

7475 The <*additional info*> field is either null or a string that starts with a blank character and contains  
 7476 no newline characters. Some systems report on the number of lines in this case.

7477 **OUTPUT FILES**  
 7478 None.

7479 **EXTENDED DESCRIPTION**  
 7480 None.

7481 **EXIT STATUS**  
 7482 The following exit values are returned:

7483 0 The files are identical.  
 7484 1 The files are different; this includes the case where one file is identical to the first part of the  
 7485 other.  
 7486 >1 An error occurred.

7487 **CONSEQUENCES OF ERRORS**  
 7488 Default.

7489 **APPLICATION USAGE**  
 7490 Although input files to *cmp* can be any type, the results might not be what would be expected on  
 7491 character special device files or on file types not described by the **XSH** specification. Since this  
 7492 specification does not specify the block size used when doing input, comparisons of character  
 7493 special files need not compare all of the data in those files.

7494 **EXAMPLES**  
 7495 None.

7496 **FUTURE DIRECTIONS**  
 7497 None.

7498 **SEE ALSO**

7499 *comm, diff.*

7500 **CHANGE HISTORY**

7501 First released in Issue 2.

7502 **Issue 4**

7503 Aligned with the ISO/IEC 9945-2: 1993 standard.

7504 **NAME**7505 `col` — filter reverse line-feeds (**LEGACY**)7506 **SYNOPSIS**7507 EX `col [-bfp $x$ ]`7508 **DESCRIPTION**

7509 The `col` utility reads from the standard input and writes to the standard output. It performs the  
 7510 line overlays implied by reverse line-feeds, and by forward and reverse half-line-feeds. Unless  
 7511 `-x` is used, all blank characters in the input will be converted to tab characters wherever possible.

7512 The ASCII control characters SO and SI are assumed by `col` to start and end text in an alternative  
 7513 character set. The character set to which each input character belongs is remembered, and on  
 7514 output SI and SO characters are generated as appropriate to ensure that each character is written  
 7515 in the correct character set.

7516 On input, the only control characters accepted are space, backspace, tab, carriage-return and  
 7517 newline characters, SI, SO, VT, reverse line-feed, forward half-line-feed and reverse half-line-  
 7518 feed. The VT character is an alternative form of full reverse line-feed, included for compatibility  
 7519 with some earlier programs of this type. The only other characters to be copied to the output are  
 7520 those that are printable.

7521 The ASCII codes for the control functions and line-motion sequences mentioned above are as  
 7522 given in the table below. ESC stands for the ASCII escape character, with the octal code 033;  
 7523 ESC- $x$  means a sequence of two characters, ESC followed by the character  $x$ .

|      |                        |       |
|------|------------------------|-------|
| 7524 | reverse line-feed      | ESC-7 |
| 7525 | reverse half-line-feed | ESC-8 |
| 7526 | forward half-line-feed | ESC-9 |
| 7527 | vertical-tab (VT)      | 013   |
| 7528 | start-of-text (SO)     | 016   |
| 7529 | end-of-text (SI)       | 017   |

7530 **OPTIONS**

7531 The `col` utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 7532 following options are supported:

- 7533 **-b** Assume that the output device in use is not capable of backspacing. In this case, if two  
 7534 or more characters are to appear in the same place, only the last one read will be  
 7535 output.
- 7536 **-f** Suppress the normal treatment of half-line motions. Although `col` accepts half-line  
 7537 motions in its input, it normally does not emit them on output. Instead, text that would  
 7538 appear between lines is moved to the next lower full-line boundary. By suppressing  
 7539 this treatment, the output from `col` may contain forward half-line-feeds, but will still  
 7540 never contain either kind of reverse-line motion.
- 7541 **-p** Force escape sequences to be passed through unchanged. Normally, `col` will remove  
 7542 any escape sequences found in its input that are not specified above.
- 7543 **-x** Prevent `col` from converting blank characters to tab characters on output wherever  
 7544 possible. Tab stops are considered to be at each column position  $n$  such that  $n$  modulo  
 7545 8 equals 1.

7546 **OPERANDS**

7547 None.

**7548 STDIN**

7549       The standard input is a text file to be translated.

**7550 INPUT FILES**

7551       None.

**7552 ENVIRONMENT VARIABLES**

7553       The following environment variables may affect the execution of *col*:

7554       *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
7555       *LANG* is unset or null, the corresponding value from the implementation-dependent  
7556       default locale will be used. If any of the internationalisation variables contains an  
7557       invalid setting, the utility will behave as if none of the variables had been defined.

7558       *LC\_ALL*

7559       If set to a non-empty string value, override the values of all the other  
7560       internationalisation variables.

7561       *LC\_CTYPE*

7562       Determine the locale for the interpretation of sequences of bytes of text data as  
7563       characters (for example, single- as opposed to multi-byte characters in arguments and  
7564       input files).

7565       *LC\_MESSAGES*

7566       Determine the locale that should be used to affect the format and contents of diagnostic  
7567       messages written to standard error.

7568       *NLSPATH*

7569       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

**7570 ASYNCHRONOUS EVENTS**

7571       Default.

**7572 STDOUT**

7573       The standard output is a text file, translated from the standard input.

**7574 STDERR**

7575       Used only for diagnostic messages.

**7576 OUTPUT FILES**

7577       None.

**7578 EXTENDED DESCRIPTION**

7579       None.

**7580 EXIT STATUS**

7581       The following exit values are returned:

7582       0   Successful completion.

7583       >0  An error occurred.

**7584 CONSEQUENCES OF ERRORS**

7585       Default.

**7586 APPLICATION USAGE**

7587       The use of the *-x* option may increase or decrease printing time, depending on the printer type.

7588       Local vertical motions that would result in backing up over the first line of the document are  
7589       ignored. As a result, the first line must not have any superscripts.



7590           The use of the **-f** or **-p** options is discouraged unless the user is aware of the consequences of  
7591           passing unusual escape sequences to the terminal.

7592 **EXAMPLES**

7593           None.

7594 **FUTURE DIRECTIONS**

7595           None.

7596 **SEE ALSO**

7597           None.

7598 **CHANGE HISTORY**

7599           First released in Issue 2.

7600 **Issue 4**

7601           Format reorganised.

7602           Internationalised environment variable support made optional.

7603           Marked TO BE WITHDRAWN.

7604 **Issue 5**

7605           Marked LEGACY.

7606 **NAME**

7607       comm — select or reject lines common to two files

7608 **SYNOPSIS**7609       comm [-123] *file1 file2*7610 **DESCRIPTION**

7611       The *comm* utility will read *file1* and *file2*, which should be ordered in the current collating  
 7612       sequence, and produce three text columns as output: lines only in *file1*; lines only in *file2*; and  
 7613       lines in both files.

7614       If the lines in both files are not ordered according to the collating sequence of the current locale,  
 7615       the results are unspecified.

7616 **OPTIONS**7617       The *comm* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

7618       The following options are supported:

7619       -1       Suppress the output column of lines unique to *file1*.7620       -2       Suppress the output column of lines unique to *file2*.7621       -3       Suppress the output column of lines duplicated in *file1* and *file2*.7622 **OPERANDS**

7623       The following operands are supported:

7624       *file1*    A pathname of the first file to be compared. If *file1* is "-", the standard input is used.7625       *file2*    A pathname of the second file to be compared. If *file2* is "-", the standard input is used.

7626       If both *file1* and *file2* refer to standard input or to the same FIFO special, block special or  
 7627       character special file, the results are undefined.

7628 **STDIN**

7629       The standard input will be used only if one of the *file1* or *file2* operands refers to standard input.  
 7630       See the INPUT FILES section.

7631 **INPUT FILES**

7632       The input files must be text files.

7633 **ENVIRONMENT VARIABLES**7634       The following environment variables affect the execution of *comm*:

7635       **LANG**    Provide a default value for the internationalisation variables that are unset or null. If  
 7636       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 7637       default locale will be used. If any of the internationalisation variables contains an  
 7638       invalid setting, the utility will behave as if none of the variables had been defined.

7639       **LC\_ALL**

7640       If set to a non-empty string value, override the values of all the other  
 7641       internationalisation variables.

7642       **LC\_COLLATE**

7643       Determine the locale for the collating sequence *comm* expects to have been used when  
 7644       the input files were sorted.

7645       **LC\_CTYPE**

7646       Determine the locale for the interpretation of sequences of bytes of text data as  
 7647       characters (for example, single- as opposed to multi-byte characters in arguments and  
 7648       input files).

7649 **LC\_MESSAGES**  
 7650 Determine the locale that should be used to affect the format and contents of diagnostic  
 7651 messages written to standard error.

7652 EX **NLSPATH**  
 7653 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

7654 **ASYNCHRONOUS EVENTS**  
 7655 Default.

7656 **STDOUT**  
 7657 The *comm* utility will produce output depending on the options selected. If the **-1**, **-2** and **-3**  
 7658 options are all selected, *comm* will write nothing to standard output.

7659 If the **-1** option is not selected, lines contained only in *file1* will be written using the format:  
 7660 "%s\n", <line in file1>

7661 If the **-2** option is not selected, lines contained only in *file2* will be written using the format:  
 7662 "%s%s\n", <lead>, <line in file2>

7663 where the string <lead> is:  
 7664 <tab> if the **-1** option is not selected, or  
 7665 null string if the **-1** option is selected.

7666 If the **-3** option is not selected, lines contained in both files will be written using the format:  
 7667 "%s%s\n", <lead>, <line in both>

7668 where the string <lead> is:  
 7669 <tab><tab> if neither the **-1** nor the **-2** option is selected, or  
 7670 <tab> if exactly one of the **-1** and **-2** options is selected, or  
 7671 null string if both the **-1** and **-2** options are selected.

7672 If the input files were ordered according to the collating sequence of the current locale, the lines  
 7673 written will be in the collating sequence of the original lines.

7674 **STDERR**  
 7675 Used only for diagnostic messages.

7676 **OUTPUT FILES**  
 7677 None.

7678 **EXTENDED DESCRIPTION**  
 7679 None.

7680 **EXIT STATUS**  
 7681 The following exit values are returned:  
 7682 0 All input files were successfully output as specified.  
 7683 >0 An error occurred.

7684 **CONSEQUENCES OF ERRORS**  
 7685 Default.

7686 **APPLICATION USAGE**  
 7687 If the input files are not properly presorted, the output of *comm* might not be useful.

7688 **EXAMPLES**

7689 If a file named **xpg4** contains a sorted list of the utilities in this specification, a file named **xpg3** |  
7690 contains a sorted list of the utilities specified in the **X/Open Portability Guide, Issue 3**, and a file  
7691 named **svid89** contains a sorted list of the utilities in the System V Interface Definition Third  
7692 Edition:

7693 `comm -23 xpg4 xpg3 | comm -23 - svid89`

7694 would print a list of utilities in this specification not specified by either of the other documents; |

7695 `comm -12 xpg4 xpg3 | comm -12 - svid89`

7696 would print a list of utilities specified by all three documents; and

7697 `comm -12 xpg3 svid89 | comm -23 - xpg4`

7698 would print a list of utilities specified by both XPG3 and the SVID, but not specified in this |  
7699 specification.

7700 **FUTURE DIRECTIONS**

7701 None.

7702 **SEE ALSO**

7703 *cmp, diff, sort, uniq.*

7704 **CHANGE HISTORY**

7705 First released in Issue 2. |

7706 **Issue 4**

7707 Aligned with the ISO/IEC 9945-2: 1993 standard.

7708 **NAME**

7709       command — execute a simple command

7710 **SYNOPSIS**7711       command [-p] *command\_name* [*argument* ...]7712       command [-v | -V ] *command\_name*7713 **DESCRIPTION**

7714       The *command* utility causes the shell to treat the arguments as a simple command, suppressing  
 7715       the shell function lookup that is described in **Command Search and Execution** on page 47 item  
 7716       1b.

7717       If the *command\_name* is the same as the name of one of the special built-in utilities, the special  
 7718       properties in the enumerated list at the beginning of Section 2.14 on page 67 will not occur. In  
 7719       every other respect, if *command\_name* is not the name of a function, the effect of *command* will be  
 7720       the same as omitting *command*.

7721       The *command* utility also provides information concerning how a command name will be  
 7722       interpreted by the shell; see -v and -V.

7723 **OPTIONS**7724       The *command* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

7725       The following options are supported:

7726       -p       Perform the command search using a default value for *PATH* that is guaranteed to find  
 7727       all of the standard utilities.

7728       -v       Write a string to standard output that indicates the pathname or command that will be  
 7729       used by the shell, in the current shell execution environment (see Section 2.12 on page  
 7730       63), to invoke *command\_name*.

7731               • Utilities, regular built-in utilities, *command\_names* including a slash character, and  
 7732               any implementation-dependent functions that are found using the *PATH* variable  
 7733               (as described in **Command Search and Execution** on page 47), will be written as  
 7734               absolute pathnames.

7735               • Shell functions, special built-in utilities, regular built-in utilities not associated with  
 7736               a *PATH* search, and shell reserved words will be written as just their names.

7737               • An alias will be written as a command line that represents its alias definition.

7738               • Otherwise, no output will be written and the exit status will reflect that the name  
 7739               was not found.

7740       -V       Write a string to standard output that indicates how the name given in the  
 7741       *command\_name* operand will be interpreted by the shell, in the current shell execution  
 7742       environment (see Section 2.12 on page 63). Although the format of this string is  
 7743       unspecified, it will indicate in which of the following categories *command\_name* falls  
 7744       and include the information stated:

7745               • Utilities, regular built-in utilities, and any implementation-dependent functions that  
 7746               are found using the *PATH* variable (as described in **Command Search and**  
 7747               **Execution** on page 47), will be identified as such and include the absolute pathname  
 7748               in the string.

7749               • Other shell functions will be identified as functions.

7750               • Aliases will be identified as aliases and their definitions will be included in the  
 7751               string.

- 7752 • Special built-in utilities will be identified as special built-in utilities.
- 7753 • Regular built-in utilities not associated with a *PATH* search will be identified as
- 7754 regular built-in utilities. (The term “regular” need not be used.)
- 7755 • Shell reserved words will be identified as reserved words.

## 7756 OPERANDS

7757 The following operands are supported:

7758 *argument*

7759 One of the strings treated as an argument to *command\_name*.

7760 *command\_name*

7761 The name of a utility or a special built-in utility.

## 7762 STDIN

7763 Not used.

## 7764 INPUT FILES

7765 None.

## 7766 ENVIRONMENT VARIABLES

7767 The following environment variables affect the execution of *command*:

7768 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 7769 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 7770 default locale will be used. If any of the internationalisation variables contains an  
 7771 invalid setting, the utility will behave as if none of the variables had been defined.

7772 *LC\_ALL*

7773 If set to a non-empty string value, override the values of all the other  
 7774 internationalisation variables.

7775 *LC\_CTYPE*

7776 Determine the locale for the interpretation of sequences of bytes of text data as  
 7777 characters (for example, single- as opposed to multi-byte characters in arguments).

7778 *LC\_MESSAGES*

7779 Determine the locale that should be used to affect the format and contents of diagnostic  
 7780 messages written to standard error and informative messages written to standard  
 7781 output.

7782 EX *NLSPATH*

7783 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

7784 *PATH* Determine the search path used during the command search described in **Command**  
 7785 **Search and Execution** on page 47, except as described under the **-p** option.

## 7786 ASYNCHRONOUS EVENTS

7787 Default.

## 7788 STDOUT

7789 When the **-v** option is specified, standard output is formatted as:

7790 "%s\n", <pathname or command>

7791 When the **-V** option is specified, standard output is formatted as:

7792 "%s\n", <unspecified>

7793 **STDERR**

7794       Used only for diagnostic messages.

7795 **OUTPUT FILES**

7796       None.

7797 **EXTENDED DESCRIPTION**

7798       None.

7799 **EXIT STATUS**7800       When the **-v** or **-V** options are specified, the following exit values are returned:

7801       0       Successful completion.

7802       >0      The *command\_name* could not be found or an error occurred.

7803       Otherwise, the following exit values are returned:

7804       126     The utility specified by *command\_name* was found but could not be invoked.7805       127     An error occurred in the *command* utility or the utility specified by *command\_name* could not be found.7807       Otherwise, the exit status of *command* will be that of the simple command specified by the arguments to *command*.7809 **CONSEQUENCES OF ERRORS**

7810       Default.

7811 **APPLICATION USAGE**7812       The order for command search allows functions to override regular built-ins and path searches.  
7813       This utility is necessary to allow functions that have the same name as a utility to call the utility  
7814       (instead of a recursive call to the function).7815       The system default path is available using *getconf*; however, since *getconf* may need to have the  
7816       *PATH* set up before it can be called itself, the following can be used:7817       `command -p getconf _CS_PATH`7818       There are some advantages to suppressing the special characteristics of special built-ins on  
7819       occasion. For example:7820       `command exec > unwritable-file`7821       will not cause a non-interactive script to abort, so that the output status can be checked by the  
7822       script.7823       The *command*, *env*, *nohup*, *time* and *xargs* utilities have been specified to use exit code 127 if an  
7824       error occurs so that applications can distinguish “failure to find a utility” from “invoked utility  
7825       exited with an error indication”. The value 127 was chosen because it is not commonly used for  
7826       other meanings; most utilities use small values for “normal error conditions” and the values  
7827       above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen  
7828       in a similar manner to indicate that the utility could be found, but not invoked. Some scripts  
7829       produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
7830       between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
7831       *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
7832       any other reason.

Since the `-v` and `-V` options of *command* produce output in relation to the current shell execution environment, *command* is generally provided as a shell regular built-in. If it is called in a subshell or separate utility execution environment, such as one of the following:

```
(PATH=foo command -v)
nohup command -v
```

it will not necessarily produce correct results. For example, when called with *nohup* or an *exec* function, in a separate utility execution environment, most implementations will not be able to identify aliases, functions or special built-ins.

Two types of regular built-ins could be encountered on a system and these are described separately by *command*. The description of command search in **Command Search and Execution** on page 47 allows for a standard utility to be implemented as a regular built-in as long as it is found in the appropriate place in a *PATH* search. So, for example, *command -v true* might yield */bin/true* or some similar pathname. Other implementation-dependent utilities that are not defined by this specification might exist only as built-ins and have no pathname associated with them. These will produce output identified as (regular) built-ins. Applications encountering these will not be able to count on *execing* them, using them with *nohup*, overriding them with a different *PATH*, and so on.

## EXAMPLES

1. Make a version of *cd* that always prints out the new working directory exactly once:

```
cd() {
 command cd "$@" >/dev/null
 pwd
}
```

2. Start off a “secure shell script” in which the script avoids being spoofed by its parent:

```
IFS= '
The preceding value should be <space><tab><newline>.
Set IFS to its default value.

\unalias -a
Unset all possible aliases.
Note that unalias is escaped to prevent an alias
being used for unalias.

unset -f command
Ensure command is not a user function.

PATH="$(command -p getconf _CS_PATH):$PATH"
Put on a reliable PATH prefix.

...
```

At this point, given correct permissions on the directories called by *PATH*, the script has the ability to ensure that any utility it calls is the intended one. It is being very cautious because it assumes that implementation extensions may be present that would allow user functions to exist when it is invoked; this capability is not specified by this specification, but it is not prohibited as an extension. For example, the *ENV* variable precedes the invocation of the script with a user startup script. Such a script could define functions to spoof the application.



7877 **FUTURE DIRECTIONS**

7878       None.

7879 **SEE ALSO**7880       *sh*, *type*.7881 **CHANGE HISTORY**

7882       First released in Issue 4.

## 7883 NAME

7884 compress — compress data

## 7885 SYNOPSIS

7886 EX `compress [-fv][-b bits][file ...]`7887 EX `compress [-cfv][-b bits][file]`

## 7888 DESCRIPTION

7889 The *compress* utility will attempt to reduce the size of the named files by using adaptive Lempel-  
 7890 Ziv coding. Except when the output is to the standard output, each file will be replaced by one  
 7891 with the extension *.Z*. If the invoking process has appropriate privileges, the ownership, modes,  
 7892 access time, and modification time of the original file are preserved. If appending the *.Z* to the  
 7893 filename would make the name exceed {NAME\_MAX} bytes, the command will fail. If no files  
 7894 are specified, the standard input will be compressed to the standard output.

## 7895 OPTIONS

7896 The *compress* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**.

7897 The following options are supported:

7898 **-b *bits*** Specify the maximum number of bits to use in a code. For a portable application, the  
 7899 *bits* argument must be:

7900  $9 \geq bits \leq 14$ 

7901 The implementation may allow *bits* values of greater than 14. The default will be 14, 15  
 7902 or 16.

7903 **-c** Cause *compress* to write to the standard output; the input file will not be changed, and  
 7904 no *.Z* files will be created.

7905 **-f** Force compression of *file*, even if it does not actually reduce the size of the file, or if the  
 7906 corresponding *file.Z* file already exists. If the **-f** option is not given, and the process is  
 7907 not running in the background, the user will be prompted as to whether an existing  
 7908 *file.Z* file should be overwritten.

7909 **-v** Write the percentage reduction of each file to standard error.

## 7910 OPERANDS

7911 The following operand is supported:

7912 *file* A pathname of a file to be compressed.

## 7913 STDIN

7914 The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".

## 7915 INPUT FILES

7916 If *file* operands are specified, the input files contain the data to be compressed.

## 7917 ENVIRONMENT VARIABLES

7918 The following environment variables affect the execution of *compress*:

7919 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 7920 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 7921 default locale will be used. If any of the internationalisation variables contains an  
 7922 invalid setting, the utility will behave as if none of the variables had been defined.

7923 **LC\_ALL**

7924 If set to a non-empty string value, override the values of all the other  
 7925 internationalisation variables.

7926        **LC\_CTYPE**  
 7927            Determine the locale for the interpretation of sequences of bytes of text data as  
 7928            characters (for example, single- as opposed to multi-byte characters in arguments).

7929        **LC\_MESSAGES**  
 7930            Determine the locale that should be used to affect the format and contents of diagnostic  
 7931            messages written to standard error.

7932        **NLSPATH**  
 7933            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

7934        **ASYNCHRONOUS EVENTS**  
 7935            Default.

7936        **STDOUT**  
 7937            If no *file* operands are specified, or if a *file* operand is "-", or if the **-c** option is specified, the  
 7938            standard output will contain the compressed output.

7939        **STDERR**  
 7940            Used for all diagnostic and prompt messages and the output from **-v**.

7941        **OUTPUT FILES**  
 7942            The output files will contain the compressed output.

7943        **EXTENDED DESCRIPTION**  
 7944            None.

7945        **EXIT STATUS**  
 7946            The following exit values are returned:

7947            0    Successful completion.  
 7948            1    An error occurred.  
 7949            2    One or more files were not compressed because they would have increased in size (and the  
 7950            **-f** option was not specified).  
 7951            >2   An error occurred.

7952        **CONSEQUENCES OF ERRORS**  
 7953            The input file will remain unmodified.

7954        **APPLICATION USAGE**  
 7955            The amount of compression obtained depends on the size of the input, the number of *bits* per  
 7956            code, and the distribution of common substrings. Typically, text such as source code or English  
 7957            is reduced by 50–60%. Compression is generally much better than that achieved by Huffman  
 7958            coding (as used in *pack*), or adaptive Huffman coding (*compact*), and takes less time to compute.

7959            Although *compress* strictly follows the default actions upon receipt of a signal or when an error  
 7960            occurs, some unexpected results may occur. In some implementations it is likely that a partially  
 7961            compressed file will be left in place, alongside its uncompressed input file. Since the general  
 7962            operation of *compress* is to delete the uncompressed file only after the *.Z* file has been  
 7963            successfully filled, an application should always carefully check the exit status of *compress* before  
 7964            arbitrarily deleting files that have like-named neighbours with *.Z* suffixes.

7965            Compressed files are not necessarily portable to other systems.

7966            The limit of 14 on the *bits* option-argument is to achieve portability to all systems (within the  
 7967            restrictions imposed by the lack of an explicit published file format). Some systems based on  
 7968            16-bit architectures cannot support 15- or 16-bit uncompression.

7969 **EXAMPLES**

7970           None.

7971 **FUTURE DIRECTIONS**

7972           None.

7973 **SEE ALSO**7974           *uncompress*, *zcat*.7975 **CHANGE HISTORY**

7976           First released in Issue 4.

7977 **Issue 4, Version 2**

7978           The following changes are made:

- 7979           • The DESCRIPTION section is clarified to state that the ownership, modes, access time and  
7980           modification time of the original file are preserved if the invoking process has appropriate  
7981           privileges.
- 7982           • The STDOUT section includes the case where a *file* operand is "-".

## 7983 NAME

7984 cp — copy files

## 7985 SYNOPSIS

7986 cp [-fip] *source\_file* *target\_file*7987 cp [-fip] *source\_file* ... *target*7988 cp -R[-fip] *source\_file* ... *target*7989 cp -r[-fip] *source\_file* ... *target*

## 7990 DESCRIPTION

7991 The first synopsis form is denoted by two operands, neither of which are existing files of type  
 7992 directory. The *cp* utility will copy the contents of *source\_file* to the destination path named by  
 7993 *target\_file*.

7994 The second synopsis form is denoted by two or more operands where the **-R** or **-r** options are  
 7995 not specified and the first synopsis form is not applicable. It is an error if any *source\_file* is a file  
 7996 of type directory, if *target* does not exist or if *target* is a file of a type defined by the **XSH**  
 7997 specification, but is not a file of type directory. The *cp* utility will copy the contents of each  
 7998 *source\_file* to the destination path named by the concatenation of *target*, a slash character and the  
 7999 last component of *source\_file*.

8000 The third and fourth synopsis forms are denoted by two or more operands where the **-R** or **-r**  
 8001 options are specified. The *cp* utility will copy each file in the file hierarchy rooted in each  
 8002 *source\_file* to a destination path named as follows.

8003 If *target* exists and is a file of type directory, the name of the corresponding destination path for  
 8004 each file in the file hierarchy will be the concatenation of *target*, a slash character and the  
 8005 pathname of the file relative to the directory containing *source\_file*.

8006 If *target* does not exist and two operands are specified, the name of the corresponding  
 8007 destination path for *source\_file* will be *target*; the name of the corresponding destination path for  
 8008 all other files in the file hierarchy will be the concatenation of *target*, a slash character and the  
 8009 pathname of the file relative to *source\_file*.

8010 It is an error if *target* does not exist and more than two operands are specified, or if *target* exists  
 8011 and is a file of a type defined by the **XSH** specification, but is not a file of type directory.

8012 In the following description, *source\_file* refers to the file that is being copied, whether specified as  
 8013 an operand or a file in a file hierarchy rooted in a *source\_file* operand. The term *dest\_file* refers to  
 8014 the file named by the destination path.

8015 For each *source\_file*, the following steps will be taken:

- 8016 1. If *source\_file* references the same file as *dest\_file*, *cp* may write a diagnostic message to  
 8017 standard error; it will do nothing more with *source\_file* and will go on to any remaining  
 8018 files.
- 8019 2. If *source\_file* is of type directory, the following steps will be taken:
  - 8020 a. If neither the **-R** or **-r** options were specified, *cp* will write a diagnostic message to  
 8021 standard error, do nothing more with *source\_file* and go on to any remaining files.
  - 8022 b. If *source\_file* was not specified as an operand and *source\_file* is dot or dot-dot, *cp* will  
 8023 do nothing more with *source\_file* and go on to any remaining files.
  - 8024 c. If *dest\_file* exists and it is a file type not specified by the **XSH** specification, the  
 8025 behaviour is implementation-dependent.

- 8026 d. If *dest\_file* exists and it is not of type directory, *cp* will write a diagnostic message to  
 8027 standard error, do nothing more with *source\_file* or any files below *source\_file* in the  
 8028 file hierarchy, and go on to any remaining files.
- 8029 e. If the directory *dest\_file* does not exist, it will be created with file permission bits set  
 8030 to the same value as those of *source\_file*, modified by the file creation mask of the user  
 8031 if the **-p** option was not specified, and then bitwise inclusively ORed with S\_IRWXU.  
 8032 If *dest\_file* cannot be created, *cp* will write a diagnostic message to standard error, do  
 8033 nothing more with *source\_file*, and go on to any remaining files. It is unspecified if *cp*  
 8034 will attempt to copy files in the file hierarchy rooted in *source\_file*.
- 8035 f. The files in the directory *source\_file* will be copied to the directory *dest\_file*, taking the  
 8036 four steps [1–4] listed here with the files as *source\_files*.
- 8037 g. If *dest\_file* was created, its file permission bits will be changed (if necessary) to be the  
 8038 same as those of *source\_file*, modified by the file creation mask of the user if the **-p**  
 8039 option was not specified.
- 8040 h. The *cp* utility will do nothing more with *source\_file* and go on to any remaining files.
- 8041 3. If *source\_file* is of type regular file, the following steps will be taken:
- 8042 a. If *dest\_file* exists, the following steps are taken:
- 8043 i. If the **-i** option is in effect, the *cp* utility will write a prompt to the standard  
 8044 error and read a line from the standard input. If the response is not affirmative,  
 8045 *cp* will do nothing more with *source\_file* and go on to any remaining files.
- 8046 ii. A file descriptor for *dest\_file* will be obtained by performing actions equivalent  
 8047 to the **XSH** specification *open()* function called using *dest\_file* as the *path*  
 8048 argument, and the bitwise inclusive OR of O\_WRONLY and O\_TRUNC as the *oflag*  
 8049 argument.
- 8050 iii. If the attempt to obtain a file descriptor fails and the **-f** option is in effect, *cp*  
 8051 will attempt to remove the file by performing actions equivalent to the **XSH**  
 8052 specification *unlink()* function called using *dest\_file* as the *path* argument. If  
 8053 this attempt succeeds, *cp* will continue with step 3b.
- 8054 b. If *dest\_file* does not exist, a file descriptor will be obtained by performing actions  
 8055 equivalent to the **XSH** specification *open()* function called using *dest\_file* as the *path*  
 8056 argument, and the bitwise inclusive OR of O\_WRONLY and O\_CREAT as the *oflag*  
 8057 argument. The file permission bits of *source\_file* will be the *mode* argument.
- 8058 c. If the attempt to obtain a file descriptor fails, *cp* will write a diagnostic message to  
 8059 standard error, do nothing more with *source\_file*, and go on to any remaining files.
- 8060 d. The contents of *source\_file* will be written to the file descriptor. Any write errors will  
 8061 cause *cp* to write a diagnostic message to standard error and continue to step 3e.
- 8062 e. The file descriptor will be closed.
- 8063 f. The *cp* utility will do nothing more with *source\_file*. If a write error occurred in step  
 8064 3d, it is unspecified if *cp* continues with any remaining files. If no write error  
 8065 occurred in step 3d, *cp* will go on to any remaining files.
- 8066 4. Otherwise, the following steps will be taken:
- 8067 a. If the **-r** option was specified, the behaviour is implementation-dependent.

- 8068           b. If the **-R** option was specified, the following steps will be taken:
- 8069           i. The *dest\_file* will be created with the same file type as *source\_file*.
- 8070           ii. If *source\_file* is a file of type FIFO, the file permission bits will be the same as
- 8071               those of *source\_file*, modified by the file creation mask of the user if the **-p**
- 8072               option was not specified. Otherwise, the permissions, owner ID and group ID
- 8073               of *dest\_file* are implementation-dependent.
- 8074               If this creation fails for any reason, *cp* will write a diagnostic message to
- 8075               standard error, do nothing more with *source\_file* and go on to any remaining
- 8076               files.

8077           If the implementation provides additional or alternate access control mechanisms (see **file**

8078           **access permissions** in the XBD specification, **Chapter 2, Glossary**), their effect on copies of files

8079           is implementation-dependent.

## 8080 OPTIONS

8081           The *cp* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

8082           The following options are supported:

- 8083           **-f**       If a file descriptor for a destination file cannot be obtained, as described in step 3.a.ii.,
- 8084               attempt to unlink the destination file and proceed.
- 8085           **-i**       Write a prompt to standard error before copying to any existing destination file. If the
- 8086               response from the standard input is affirmative, the copy will be attempted, otherwise
- 8087               not.
- 8088           **-p**       Duplicate the following characteristics of each source file in the corresponding
- 8089               destination file:
- 8090               1. The time of last data modification and time of last access. If this duplication fails
- 8091                 for any reason, *cp* will write a diagnostic message to standard error.
- 8092               2. The user ID and group ID. If this duplication fails for any reason, it is unspecified
- 8093                 whether *cp* writes a diagnostic message to standard error.
- 8094               3. The file permission bits and the S\_ISUID and S\_ISGID bits. Other,
- 8095                 implementation-dependent, bits may be duplicated as well. If this duplication
- 8096                 fails for any reason, *cp* will write a diagnostic message to standard error.

8097           If the user ID or the group ID cannot be duplicated, the file permission bits S\_ISUID

8098           and S\_ISGID will be cleared. If these bits are present in the source file but are not

8099           duplicated in the destination file, it is unspecified whether *cp* writes a diagnostic

8100           message to standard error.

8101           The order in which the preceding characteristics are duplicated is unspecified. The

8102           *dest\_file* will not be deleted if these characteristics cannot be preserved.

8103           **-R**       Copy file hierarchies.

8104           **-r**       Copy file hierarchies. The treatment of special files is implementation-dependent.

## 8105 OPERANDS

8106           The following operands are supported:

8107           *source\_file*

8108               A pathname of a file to be copied.

8109           *target\_file*  
8110           A pathname of an existing or non-existing file, used for the output when a single file is  
8111           copied.

8112           *target*   A pathname of a directory to contain the copied files.

8113 **STDIN**  
8114           Used to read an input line in response to each prompt specified in the STDERR section.  
8115           Otherwise, the standard input will not be used.

8116 **INPUT FILES**  
8117           The input files specified as operands may be of any file type.

8118 **ENVIRONMENT VARIABLES**  
8119           The following environment variables affect the execution of *cp*:

8120           *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
8121           *LANG* is unset or null, the corresponding value from the implementation-dependent  
8122           default locale will be used. If any of the internationalisation variables contains an  
8123           invalid setting, the utility will behave as if none of the variables had been defined.

8124           *LC\_ALL*  
8125           If set to a non-empty string value, override the values of all the other  
8126           internationalisation variables.

8127           *LC\_COLLATE*  
8128           Determine the locale for the behaviour of ranges, equivalence classes and multi-  
8129           character collating elements used in the extended regular expression defined for the  
8130           **yesexpr** locale keyword in the LC\_MESSAGES category.

8131           *LC\_CTYPE*  
8132           Determine the locale for the interpretation of sequences of bytes of text data as  
8133           characters (for example, single- as opposed to multi-byte characters in arguments and  
8134           input files) and the behaviour of character classes used in the extended regular  
8135           expression defined for the **yesexpr** locale keyword in the LC\_MESSAGES category.

8136           *LC\_MESSAGES*  
8137           Determine the locale for the processing of affirmative responses that should be used to  
8138           affect the format and contents of diagnostic messages written to standard error.

8139 EX       *NLSPATH*  
8140           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

8141 **ASYNCHRONOUS EVENTS**  
8142           Default.

8143 **STDOUT**  
8144           Not used.

8145 **STDERR**  
8146           A prompt will be written to standard error under the conditions specified in the DESCRIPTION  
8147           section. The prompt will contain the destination pathname, but its format is otherwise  
8148           unspecified. Otherwise, the standard error will be used only for diagnostic messages.

8149 **OUTPUT FILES**  
8150           The output files may be of any type.

8151 **EXTENDED DESCRIPTION**  
8152           None.



**8153 EXIT STATUS**

8154 The following exit values are returned:

8155 0 All files were copied successfully.

8156 >0 An error occurred.

**8157 CONSEQUENCES OF ERRORS**

8158 If *cp* is prematurely terminated by a signal or error, files or file hierarchies may be only partially  
8159 copied and files and directories may have incorrect permissions or access and modification  
8160 times.

**8161 APPLICATION USAGE**

8162 The difference between **-R** and **-r** is in the treatment by *cp* of file types other than regular and  
8163 directory. The original **-r** flag, for historic reasons, does not handle special files any differently  
8164 from regular files, but always reads the file and copies its contents. This has obvious problems  
8165 in the presence of special file types, for example character devices, FIFOs and sockets. The **-R**  
8166 option is intended to recreate the file hierarchy and the **-r** option supports historical practice. It  
8167 is anticipated that a future issue of this specification will deprecate the **-r** option, and for that  
8168 reason, there has been no attempt to fix its behaviour with respect to FIFOs or other file types  
8169 where copying the file is clearly wrong. However, some systems support **-r** with the same  
8170 abilities as the **-R** defined in the ISO/IEC 9945-2: 1993 standard. To accommodate them as well  
8171 as systems that do not, the differences between **-r** and **-R** are implementation-dependent.  
8172 Implementations may make them identical.

8173 The set-user-ID and set-group-ID bits are explicitly cleared when files are created. This is to  
8174 prevent users from creating programs that are set-user-ID or set-group-ID to them when  
8175 copying files or to make set-user-ID or set-group-ID files accessible to new groups of users. For  
8176 example, if a file is set-user-ID and the copy has a different group ID than the source, a new  
8177 group of users has execute permission to a set-user-ID program than did previously. In  
8178 particular, this is a problem for superusers copying users' trees.

**8179 EXAMPLES**

8180 None.

**8181 FUTURE DIRECTIONS**

8182 The **-r** option may be removed; use **-R** instead.

**8183 SEE ALSO**

8184 *mv, find, ln, pax.*

**8185 CHANGE HISTORY**

8186 First released in Issue 2.

**8187 Issue 4**

8188 Aligned with the ISO/IEC 9945-2: 1993 standard.

8189 **NAME**

8190 cpio — copy file archives in and out (**LEGACY**)

8191 **SYNOPSIS**

8192 EX `cpio -o[aBcv]`

8193 EX `cpio -i[Bcdmrtuvf] [pattern ...]`

8194 EX `cpio -p[adlmuv] directory`

8195 **DESCRIPTION**

8196 The *cpio* utility, depending on the options used:

- 8197 • copies files to an archive file
- 8198 • extracts files from an archive file
- 8199 • copies files from one directory tree to another.

8200 **OPTIONS**

8201 The *cpio* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
8202 the option modifiers cannot be presented as separate arguments from the option letters.

8203 The following options are supported:

8204 **-o** (Copy Out.) Read the standard input to obtain a list of pathnames and copy those files  
8205 onto the standard output together with pathname and status information. Output is  
8206 padded to a 512-byte boundary.

8207 **-i** (Copy In.) Extract files from the standard input, which is assumed to be the product of  
8208 a previous *cpio -o*. Only files with names that match *patterns* are selected. The  
8209 extracted files are conditionally created and copied into the current directory tree based  
8210 upon the options described below. The permissions of the files will be those of the  
8211 previous *cpio -o*. The owner and group of the files will be that of the current user  
8212 unless the user has appropriate privileges, which causes *cpio* to retain the owner and  
8213 group of the files of the previous *cpio -o*. If the archive being read does not match the  
8214 modifier specified, *cpio* may consider this to be an error and exit or may recognise the  
8215 archive and continue processing. Only a user with appropriate privileges can extract  
8216 block special or character special files from an archive.

8217 **-p** (Pass.) Read the standard input to obtain a list of pathnames of files that are  
8218 conditionally created and copied into the destination *directory* tree based upon the  
8219 option modifiers described below.

8220 The following option modifiers can be appended in any sequence to the **-o**, **-i** or **-p** options:

8221 **a** Reset access times of input files after they have been copied. (When option **l** (see  
8222 below) is also specified, the access times of the linked files are not reset.)

8223 **B** Block input/output 5120 bytes to the record (does not apply to the **-p** option;  
8224 meaningful only with data directed to or from character special files).

8225 **d** Create directories as needed.

8226 **c** Write or read header information in character form for portability.

8227 **r** Interactively rename files. For each archive member matching *pattern* operand, a  
8228 prompt will be written to the file **/dev/tty**. The prompt will contain the name of the  
8229 archive member, but the format is otherwise unspecified. A line will then be read from  
8230 **/dev/tty**. If this line is blank, the archive member will be skipped. If this line consists of  
8231 a single period, the archive member will be processed with no modification to its name.

8232 Otherwise, its name will be replaced with the contents of the line. The *cpio* utility will  
 8233 immediately exit with a non-zero exit status if end-of-file is encountered when reading a  
 8234 response, or if **/dev/tty** cannot be opened for reading and writing.

8235 **t** Write a table of contents of the input. No files are created.

8236 **u** Copy unconditionally (normally, an older file will not replace a newer file with the  
 8237 same name).

8238 **v** Verbose: print the names of the affected files. With the **t** option, provides a detailed  
 8239 listing.

8240 **l** Whenever possible, link files rather than copying them. Usable only with the **-p**  
 8241 option.

8242 **m** Retain previous file modification time. This option is ineffective on directories that are  
 8243 being copied.

8244 **f** Copy in all files except those in *patterns*.

## 8245 OPERANDS

8246 The following operands are supported:

8247 *directory*  
 8248 A pathname of an existing directory to be used as the target of *cpio -p*.

8249 *pattern* Expressions making use of a pattern-matching notation similar to that used by the shell  
 8250 for filename pattern matching, and similar to regular expressions. The following  
 8251 metacharacters are defined:

8252 \* Matches any string, including the empty string.

8253 ? Matches any single character.

8254 [...] Matches any one of the enclosed characters. A pair of characters separated by  
 8255 '-' matches any symbol between the pair (inclusive), as defined by the system  
 8256 default collating sequence. If the first character following the opening '[' is a  
 8257 '!', the results are unspecified.

8258 In *pattern*, the special characters "?", "\*" and "[" also match the "/" character. Multiple  
 8259 cases of *pattern* can be specified and if no *pattern* is specified, the default for *pattern* is "\*" (that is, select all files).

## 8261 STDIN

8262 When the **-o** or **-p** options are used, the standard input is a text file containing a list of  
 8263 pathnames, one per line, to be copied.

8264 When the **-i** option is used, the standard input is an archive file formatted as specified by *pax*  
 8265 with the **-x cpio** option.

## 8266 INPUT FILES

8267 The files identified by the pathnames in the standard input are of any type.

8268 When the **-r** option is used, the file **/dev/tty** is used to write prompts and read responses.

## 8269 ENVIRONMENT VARIABLES

8270 The following environment variables may affect the execution of *cpio*:

8271 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 8272 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 8273 default locale will be used. If any of the internationalisation variables contains an  
 8274 invalid setting, the utility will behave as if none of the variables had been defined.

8275 **LC\_ALL**  
 8276 If set to a non-empty string value, override the values of all the other  
 8277 internationalisation variables.

8278 **LC\_COLLATE**  
 8279 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 8280 character collating elements within bracketed filename patterns.

8281 **LC\_CTYPE**  
 8282 Determine the locale for the interpretation of sequences of bytes of text data as  
 8283 characters (for example, single- as opposed to multi-byte characters in arguments and  
 8284 input files) and the behaviour of character classes within bracketed filename patterns  
 8285 (for example, '[:lower:]\*').

8286 **LC\_MESSAGES**  
 8287 Determine the locale that should be used to affect the format and contents of diagnostic  
 8288 messages written to standard error.

8289 **LC\_TIME**  
 8290 Determine the format of date and time strings output when listing the contents of an  
 8291 archive with the **-v** option; for example:  
 8292 `cpio -icvt < /dev/sctmtm0`

8293 **NLSPATH**  
 8294 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

8295 **TZ** Determine the timezone used with date and time strings.

8296 **ASYNCHRONOUS EVENTS**  
 8297 Default.

8298 **STDOUT**  
 8299 When the **-o** option is used, the standard output is an archive file formatted as specified by *pax*  
 8300 with the **-x cpio** option. Otherwise, the standard output contains commentary in an  
 8301 unspecified format concerning the progress of the execution.

8302 **STDERR**  
 8303 When the **-o** option is not used, the standard error contains commentary in an unspecified  
 8304 format concerning the progress of the execution. Otherwise, the standard error is used only for  
 8305 diagnostic messages.

8306 **OUTPUT FILES**  
 8307 Output files are created, as specified by the archive, when the **-i** or **-p** options are used.

8308 **EXTENDED DESCRIPTION**  
 8309 None.

8310 **EXIT STATUS**  
 8311 The following exit values are returned:  
 8312 0 Successful completion.  
 8313 >0 An error occurred.

8314 **CONSEQUENCES OF ERRORS**  
 8315 If a file or directory cannot be created or overwritten, *cpio* continues with the next file in the  
 8316 archive or file to be added to the archive.

8317 **APPLICATION USAGE**  
 8318 Archives created by *cpio* are portable between XSI-conformant systems provided the same

- 8319 procedures are used.
- 8320 The shell metacharacter notation is not fully compatible with that used by the shell and the *pax*  
 8321 utility. Not all systems support the use of the negation character [! ...] in *cpio* patterns. Portable  
 8322 applications must avoid the use of this notation.
- 8323 For portable communication of data between XSI-conformant systems, it is recommended that  
 8324 only characters defined in the ISO/IEC 646:1991 standard International Reference Version  
 8325 (equivalent to ASCII) 7-bit range of characters be used and that only characters defined in the  
 8326 Portable Filename Character Set be used for naming files. This recommendation is given  
 8327 because XSI-conformant systems support diverse codesets and run in various geographical areas  
 8328 and there is no single, well-established codeset that incorporates all of the characters of the  
 8329 languages of the various geographical areas.
- 8330 The *cpio* format only supports file sizes up to 8 gigabytes.
- 8331 Applications should migrate to the *pax* utility.
- 8332 **EXAMPLES**
- 8333 1. Copy the contents of a directory onto an archive:  
 8334 `ls | cpio -oc >../cpio.out`
- 8335 2. Duplicate a directory hierarchy:  
 8336 `cd olddir`  
 8337 `find . -depth -print | cpio -pd ../newdir`
- 8338 **FUTURE DIRECTIONS**  
 8339 None.
- 8340 **SEE ALSO**  
 8341 *ar, find, ls, pax, tar.*
- 8342 **CHANGE HISTORY**  
 8343 First released in Issue 2.
- 8344 **Issue 4**  
 8345 Format reorganised.  
 8346 Exceptions to Utility Syntax Guidelines conformance noted.  
 8347 Internationalised environment variable support made optional.  
 8348 Marked TO BE WITHDRAWN.
- 8349 **Issue 5**  
 8350 A note is added to the APPLICATION USAGE section indicating that *cpio* will only archive files  
 8351 up to 8 gigabytes in size.  
 8352 Marked LEGACY.

## 8353 NAME

8354 crontab — schedule periodic background work

## 8355 SYNOPSIS

8356 crontab [*file*]

8357 crontab [ -e | -l | -r ]

## 8358 DESCRIPTION

8359 The *crontab* utility creates, replaces or edits a user's *crontab* entry; a crontab entry is a list of  
 8360 commands and the times at which they are to be executed. The new crontab entry can be input  
 8361 by specifying *file* or input from standard input if no *file* operand is specified, or by using an  
 8362 editor, if **-e** is specified.

8363 Upon execution of a command from a crontab entry, the implementation will supply a default  
 8364 environment, defining at least the following environment variables:

8365 **HOME** A pathname of the user's home directory.

8366 **LOGNAME**

8367 The user's login name.

8368 **PATH** A string representing a search path guaranteed to find all of the standard utilities.

8369 **SHELL** A pathname of the command interpreter. When *crontab* is invoked as specified by this  
 8370 specification, the value will be a pathname for *sh*.

8371 The values of these variables when *crontab* is invoked as specified by this specification will not  
 8372 affect the default values provided when the scheduled command is run.

8373 If standard output and standard error are not redirected by commands executed from the  
 8374 crontab entry, any generated output or errors will be mailed, via an implementation-dependent  
 8375 method, to the user.

8376 EX Users are permitted to use *crontab* if their names appear in the file **/usr/lib/cron/cron.allow**. If  
 8377 that file does not exist, the file **/usr/lib/cron/cron.deny** is checked to determine if the user should  
 8378 be denied access to *crontab*. If neither file exists, only a process with appropriate privileges is  
 8379 allowed to submit a job. If only **cron.deny** exists and is empty, global usage is permitted. The  
 8380 **cron.allow** and **cron.deny** files consist of one user name per line.

## 8381 OPTIONS

8382 The *crontab* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

8383 The following options are supported:

8384 **-e** Edit a copy of the invoking user's crontab entry, or create an empty entry to edit if the  
 8385 crontab entry does not exist. When editing is complete, the entry will be installed as  
 8386 the user's crontab entry.

8387 **-l** (The letter ell.) List the invoking user's crontab entry.

8388 **-r** Remove the invoking user's crontab entry.

## 8389 OPERANDS

8390 The following operand is supported:

8391 **file** The pathname of a file that contains specifications, in the format defined in the INPUT  
 8392 FILES section, for crontab entries.

## 8393 STDIN

8394 See the INPUT FILES section.

8395 **INPUT FILES**

8396 In the POSIX locale, a crontab entry must be a text file consisting of lines of six fields each. The  
 8397 fields must be separated by blank characters. The first five fields must be integer patterns that  
 8398 specify the following:

- 8399 1. Minute (0–59)
- 8400 2. Hour (0–23)
- 8401 3. Day of the month (1–31)
- 8402 4. Month of the year (1–12)
- 8403 5. Day of the week (0–6 with 0=Sunday).

8404 Each of these patterns can be either an asterisk (meaning all valid values), an element or a list of  
 8405 elements separated by commas. An element must be either a number or two numbers separated  
 8406 by a hyphen (meaning an inclusive range). The specification of days can be made by two fields  
 8407 (day of the month and day of the week). If month, day of month and day of week are all  
 8408 asterisks, every day will be matched. If either the month or day of month is specified as an  
 8409 element or list, but the day of week is an asterisk, the month and day of month fields will specify  
 8410 the days that match. If both month and day of month are specified as asterisk, but day of week  
 8411 is an element or list, then only the specified days of the week will match. Finally, if either the  
 8412 month or day of month is specified as an element or list, and the day of week is also specified as  
 8413 an element or list, then any day matching either the month and day of month or the day of week,  
 8414 will be matched.

8415 The sixth field of a line in a crontab entry is a string that will be executed by *sh* at the specified  
 8416 times. A percent sign character in this field will be translated to a newline character. Any  
 8417 character preceded by a backslash (including the %) causes that character to be treated literally.  
 8418 Only the first line (up to a "%" or end-of-line) of the command field will be executed by the  
 8419 command interpreter. The other lines will be made available to the command as standard input.

8420 Blank lines and those whose first non-blank character is "#" will be ignored.

8421 **EX** The text files `/usr/lib/cron/cron.allow` and `/usr/lib/cron/cron.deny` contain user names, one per  
 8422 line, of users who are, respectively, authorised or denied access to the service underlying the  
 8423 *crontab* utility.

8424 **ENVIRONMENT VARIABLES**

8425 The following environment variables affect the execution of *crontab*:

8426 **EDITOR**

8427 Determine the editor to be invoked when the `-e` option is specified. The default editor  
 8428 is *vi*.

8429 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 8430 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 8431 default locale will be used. If any of the internationalisation variables contains an  
 8432 invalid setting, the utility will behave as if none of the variables had been defined.

8433 **LC\_ALL**

8434 If set to a non-empty string value, override the values of all the other  
 8435 internationalisation variables.

8436 **LC\_CTYPE**

8437 Determine the locale for the interpretation of sequences of bytes of text data as  
 8438 characters (for example, single- as opposed to multi-byte characters in arguments and  
 8439 input files).

8440 **LC\_MESSAGES**  
 8441 Determine the locale that should be used to affect the format and contents of diagnostic  
 8442 messages written to standard error.

8443 EX **NLSPATH**  
 8444 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

8445 **ASYNCHRONOUS EVENTS**  
 8446 Default.

8447 **STDOUT**  
 8448 If the **-l** option is specified, the crontab entry will be written to the standard output.

8449 **STDERR**  
 8450 Used only for diagnostic messages.

8451 **OUTPUT FILES**  
 8452 None.

8453 **EXTENDED DESCRIPTION**  
 8454 None.

8455 **EXIT STATUS**  
 8456 The following exit values are returned:  
 8457 0 Successful completion.  
 8458 >0 An error occurred.

8459 **CONSEQUENCES OF ERRORS**  
 8460 The user's crontab entry is not submitted, removed, edited or listed.

8461 **APPLICATION USAGE**  
 8462 The format of the *crontab* entry shown here is guaranteed only for the POSIX locale. Other  
 8463 cultures may be supported with substantially different interfaces, although implementations are  
 8464 encouraged to provide comparable levels of functionality.

8465 The default settings of the *HOME*, *LOGNAME*, *PATH* and *SHELL* variables that are given to the  
 8466 scheduled job are not affected by the settings of those variables when *crontab* is run; as stated,  
 8467 they are defaults. The text about "invoked as specified by this specification" means that the  
 8468 implementation may provide extensions that allow these variables to be affected at runtime, but  
 8469 that the user has to take explicit action in order to access the extension, such as give a new  
 8470 option flag or modify the format of the crontab entry.

8471 A typical user error is to type only *crontab*; this will cause the system to wait for the new crontab  
 8472 entry on standard input. If end-of-file is typed (generally <control>-D), the crontab entry will be  
 8473 replaced by an empty file. In this case, the user should type the interrupt character, which will  
 8474 prevent the crontab entry from being replaced.

8475 **EXAMPLES**

8476 1. Clean up **core** files every weekday morning at 3:15 am:  
 8477 15 3 \* \* 1-5 find \$HOME -name core 2>/dev/null | xargs rm -f

8478 2. Mail a birthday greeting:  
 8479 0 12 14 2 \* mailx john%Happy Birthday!%Time for lunch.



8480           3. As an example of specifying the two types of days:  
8481                 0 0 1,15 \* 1  
8482                 would run a command on the first and fifteenth of each month, as well as on every  
8483                 Monday. To specify days by only one field, the other field should be set to "\*"; for example:  
8484                 0 0 \* \* 1  
8485                 would run a command only on Mondays.  
8486 **FUTURE DIRECTIONS**  
8487                 None.  
8488 **SEE ALSO**  
8489                 *at*.  
8490 **CHANGE HISTORY**  
8491                 First released in Issue 2.  
8492 **Issue 4**  
8493                 Aligned with the ISO/IEC 9945-2: 1993 standard.

8494 **NAME**  
8495       csplit — split files based on context

8496 **SYNOPSIS**  
8497       csplit [-ks][-f *prefix*][-n *number*] *file* *arg1* ...*argn*

8498 **DESCRIPTION**  
8499       The *csplit* utility reads the file named by the *file* operand, writes all or part of that file into other  
8500       files as directed by the *arg* operands, and writes the sizes of the files.

8501 **OPTIONS**  
8502       The *csplit* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.  
8503       The following options are supported:

8504       -f *prefix*  
8505               Name the created files *prefix00*, *prefix01*, ..., *prefixn*. The default is **xx00** ... **xxn**. If the  
8506       *prefix* argument would create a filename exceeding {NAME\_MAX} bytes, an error will  
8507       result, *csplit* will exit with a diagnostic message and no files will be created.

8508       -k       Leave previously created files intact. By default, *csplit* will remove created files if an  
8509       error occurs.

8510       -n *number*  
8511               Use *number* decimal digits to form filenames for the file pieces. The default is 2.

8512       -s       Suppress the output of file size messages.

8513 **OPERANDS**  
8514       The following operands are supported:

8515       *file*       The pathname of a text file to be split. If *file* is "-", the standard input will be used.

8516       The operands *arg1* ... *argn* can be a combination of the following:

8517       /*regexp*/[*offset*]  
8518               Create a file using the content of the lines from the current line up to, but not including,  
8519       the line that results from the evaluation of the regular expression with *offset*, if any,  
8520       applied. The regular expression *regexp* must follow the rules for basic regular expressions  
8521       described in the XBD specification, **Section 7.3, Basic Regular Expressions**. The  
8522       optional *offset* must be a positive or negative integer value representing a number of  
8523       lines. The integer value must be preceded by "+" or "-". If the selection of lines from an  
8524       offset expression of this type would create a file with zero lines, or one with greater  
8525       than the number of lines left in the input file, the results are unspecified. After the  
8526       section is created, the current line will be set to the line that results from the evaluation  
8527       of the regular expression with any offset applied. The pattern match of *regexp* always is  
8528       applied from the current line to the end of the file.

8529       %*regexp*%[*offset*]  
8530               This operand is the same as /*regexp*/[*offset*], except that no file will be created for the  
8531       selected section of the input file.

8532       *line\_no*    Create a file from the current line up to (but not including) the line number *line\_no*.  
8533       Lines in the file will be numbered starting at one. The current line becomes *line\_no*.

8534       {*num*}      Repeat operand. This operand can follow any of the operands described previously. If  
8535       it follows a *regexp* type operand, that operand will be applied *num* more times. If it  
8536       follows a *line\_no* operand, the file will be split every *line\_no* lines, *num* times, from that  
8537       point.

8538           An error will be reported if an operand does not reference a line between the current position  
 8539           and the end of the file.

8540 **STDIN**

8541           See the INPUT FILES section.

8542 **INPUT FILES**

8543           The input file must be a text file.

8544 **ENVIRONMENT VARIABLES**

8545           The following environment variables affect the execution of *csplit*:

8546       *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 8547       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 8548       default locale will be used. If any of the internationalisation variables contains an  
 8549       invalid setting, the utility will behave as if none of the variables had been defined.

8550       *LC\_ALL*

8551           If set to a non-empty string value, override the values of all the other  
 8552           internationalisation variables.

8553       *LC\_COLLATE*

8554           Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 8555           character collating elements within regular expressions.

8556       *LC\_CTYPE*

8557           Determine the locale for the interpretation of sequences of bytes of text data as  
 8558           characters (for example, single- as opposed to multi-byte characters in arguments and  
 8559           input files) and the behaviour of character classes within regular expressions.

8560       *LC\_MESSAGES*

8561           Determine the locale that should be used to affect the format and contents of diagnostic  
 8562           messages written to standard error.

8563 **EX**       *NLSPATH*

8564           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

8565 **ASYNCHRONOUS EVENTS**

8566           If the **-k** option is specified, created files will be retained. Otherwise the default action occurs.

8567 **STDOUT**

8568           Unless the **-s** option is used, the standard output will consist of one line per file created, with a  
 8569           format as follows:

8570           "%d\n", <file size in bytes>

8571 **STDERR**

8572           Used only for diagnostic messages.

8573 **OUTPUT FILES**

8574           The output files will contain portions of the original input file, otherwise unchanged.

8575 **EXTENDED DESCRIPTION**

8576           None.

8577 **EXIT STATUS**

8578           The following exit values are returned:

8579       0   Successful completion.

8580       >0   An error occurred.

8581 **CONSEQUENCES OF ERRORS**

8582 By default, created files will be removed if an error occurs. When the **-k** option is specified,  
 8583 created files will not be removed if an error occurs.

8584 **APPLICATION USAGE**

8585 None.

8586 **EXAMPLES**

8587 1. This example creates four files, **cobol00 ... cobol03**:

8588 `csplit -f cobol file '/procedure division/' /par5./ /par16./`

8589 After editing the split files, they can be recombined as follows:

8590 `cat cobol0[0-3] > file`

8591 Note that this example overwrites the original file.

8592 2. This example would split the file after the first 99 lines, and every 100 lines thereafter, up  
 8593 to 9 999 lines; this is because lines in the file are numbered from 1 rather than zero, for  
 8594 historical reasons:

8595 `csplit -k file 100 {99}`

8596 3. Assuming that **prog.c** follows the C-language coding convention of ending routines with a  
 8597 `"}` at the beginning of the line, this example will create a file containing each separate C  
 8598 routine (up to 21) in **prog.c**:

8599 `csplit -k prog.c '%main(%' '/^}/+1' {20}`

8600 **FUTURE DIRECTIONS**

8601 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 8602 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 8603 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 8604 finalised.

8605 **SEE ALSO**

8606 *sed, split.*

8607 **CHANGE HISTORY**

8608 First released in Issue 2.

8609 **Issue 4**

8610 Aligned with the ISO/IEC 9945-2: 1993 standard.

8611 **Issue 5**

8612 FUTURE DIRECTIONS section added.

8613 **NAME**8614 `ctags` — create a tags file (**DEVELOPMENT, FORTRAN**)8615 **SYNOPSIS**8616 `ctags [-a][-f tagsfile] pathname ...`8617 `ctags -x pathname ...`8618 **DESCRIPTION**

8619 The *ctags* utility creates a *tags* file or an index of objects from C-language or FORTRAN source  
 8620 files specified by the *pathname* operands. (FORTRAN source is processed only on systems  
 8621 supporting the FORTRAN option.) The tags file lists the locators of language-specific objects  
 8622 within the source files. A locator consists of a name, pathname and either basic regular  
 8623 expression or a line number that can be used in searching for the object definition. The objects  
 8624 that will be recognised are specified in the EXTENDED DESCRIPTION section.

8625 **OPTIONS**8626 The *ctags* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

8627 The following options are supported:

8628 **-a** Append to tags file.8629 **-f *tagsfile***

8630 Write the object locator lists into *tagsfile* instead of the default file named **tags** in the  
 8631 current directory.

8632 **-x** Produce a list of object names, the line number and filename in which each is defined,  
 8633 as well as the text of that line, and write this to the standard output. A **tags** file is not  
 8634 created when **-x** is specified.

8635 **OPERANDS**8636 The following *pathname* operands are supported:

8637 ***file.c*** Files with basenames ending with the **.c** suffix are treated as C-language source code.  
 8638 Such files that are not valid input to *c89* produce unspecified results.

8639 ***file.h*** Files with basenames ending with the **.h** suffix are treated as C-language source code.  
 8640 Such files that are not valid input to *c89* produce unspecified results.

8641 ***file.f*** Files with basenames ending with the **.f** suffix are treated as FORTRAN-language  
 8642 source code. Such files that are not valid input to *fort77* produce unspecified results.

8643 The handling of other files is implementation-dependent.

8644 **STDIN**

8645 See the INPUT FILES section.

8646 **INPUT FILES**

8647 The input files must be text files containing source code in the language indicated by the  
 8648 operand filename suffixes.

8649 **ENVIRONMENT VARIABLES**8650 The following environment variables affect the execution of *ctags*:

8651 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 8652 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 8653 default locale will be used. If any of the internationalisation variables contains an  
 8654 invalid setting, the utility will behave as if none of the variables had been defined.

8655 **LC\_ALL**  
 8656 If set to a non-empty string value, override the values of all the other  
 8657 internationalisation variables.

8658 **LC\_COLLATE**  
 8659 Determine the order in which output is sorted for the **-x** option. The POSIX locale  
 8660 determines the order in which the tags file is written.

8661 **LC\_CTYPE**  
 8662 Determine the locale for the interpretation of sequences of bytes of text data as  
 8663 characters (for example, single- as opposed to multi-byte characters in arguments and  
 8664 input files). When processing C-language source code, if the locale is not compatible  
 8665 with the C locale described by the ISO C standard, the results are unspecified.

8666 **LC\_MESSAGES**  
 8667 Determine the locale that should be used to affect the format and contents of diagnostic  
 8668 messages written to standard error.

8669 EX **NLSPATH**  
 8670 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

8671 **ASYNCHRONOUS EVENTS**  
 8672 Default.

8673 **STDOUT**  
 8674 The list of object name information produced by the **-x** option is written to standard output in  
 8675 the following format:  
 8676 "%s %d %s %s", <object-name>, <line-number>, <filename>, <text>  
 8677 where <text> is the text of line <line-number> of file <filename>.

8678 **STDERR**  
 8679 Used only for diagnostic messages.

8680 **OUTPUT FILES**  
 8681 When the **-x** option is not specified, the format of the output file is:  
 8682 "%s\t%s\t/%s/\n", <identifier>, <filename>, <rexp>  
 8683 where <rexp> is a basic regular expression (see the **XBD** specification, **Section 7.3, Basic Regular**  
 8684 **Expressions**) that could be used by an editor to find the defining instance of <identifier> in  
 8685 <filename> (where "defining instance" is indicated by the declarations listed in the EXTENDED  
 8686 DESCRIPTION section).  
 8687 An alternative format is:  
 8688 "%s\t%s\t%d\n", <identifier>, <filename>, <lineno>  
 8689 where <lineno> is a decimal line number that could be used by an editor to find <identifier> in  
 8690 <filename>. This alternative format is not produced by **ctags** when it is used as described by this  
 8691 specification, but the standard utilities that process tags files are able to process this format as  
 8692 well as the preceding one.  
 8693 In either format, the file will be sorted by identifier, based on the collation sequence in the POSIX  
 8694 locale.

8695 **EXTENDED DESCRIPTION**

8696 If the operand identifies C-language source, the *ctags* utility will attempt to produce an output  
 8697 line for each of the following objects:

- 8698 • function definitions
- 8699 • type definitions
- 8700 • macros with arguments.

8701 It may also produce output for any of the following objects:

- 8702 • function prototypes
- 8703 • structures
- 8704 • unions
- 8705 • global variable definitions
- 8706 • enumeration types
- 8707 • macros without arguments
- 8708 • **#define** statements
- 8709 • **#line** statements.

8710 Any **#if** and **#ifdef** statements will produce no output. The tag **main** is treated specially in C  
 8711 programs. The tag formed is created by prefixing **M** to the name of the file, with the trailing **.c**,  
 8712 and leading pathname components (if any) removed. This special treatment of **main** makes the  
 8713 use of *ctags* practical in directories with more than one program.

8714 If the operand identifies FORTRAN source, the *ctags* utility will produce an output line for each  
 8715 function definition. It may also produce output for any of the following objects:

- 8716 • subroutine definitions
- 8717 • COMMON statements
- 8718 • PARAMETER statements
- 8719 • DATA and BLOCK DATA statements
- 8720 • statement numbers.

8721 On systems that do not support the FORTRAN option, *ctags* produces unspecified results for  
 8722 FORTRAN source code files. It writes to standard error a message identifying this condition and  
 8723 causes a non-zero exit status to be produced.

8724 It is implementation-dependent what other objects (including duplicate identifiers) produce  
 8725 output.

8726 **EXIT STATUS**

8727 The following exit values are returned:

- 8728 0 Successful completion.
- 8729 >0 An error occurred.

8730 **CONSEQUENCES OF ERRORS**

8731 Default.

8732 **APPLICATION USAGE**

8733       The output with **-x** is meant to be a simple index that can be written out as an off-line readable  
8734       function index. If the input files to *ctags* (such as *.c* files) were not created using the same locales  
8735       as those in effect when *ctags -x* is run, results might not be as expected.

8736       The description of C-language processing says “will attempt to” because the C language can be  
8737       greatly confused, especially through the use of **#defines**, and this utility would be of no use if  
8738       the real C preprocessor were run to identify them. The output from *ctags* may be fooled and  
8739       incorrect for various constructs.

8740 **EXAMPLES**

8741       None.

8742 **FUTURE DIRECTIONS**

8743       The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
8744       interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
8745       corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
8746       finalised.

8747 **SEE ALSO**

8748       *c89*, *fort77*, *vi*.

8749 **CHANGE HISTORY**

8750       First released in Issue 4.

8751 **Issue 5**

8752       FUTURE DIRECTIONS section added.



8753 **NAME**8754 `cu` — call another system (**LEGACY**)8755 **SYNOPSIS**8756 UN EX `cu -n[-dht] [-o | -e ][-l line][-s speed]`8757 UN EX `cu [-dht] [-o | -e ][-l line][-s speed] telno`8758 UN EX `cu [-dht] [-o | -e ][-s speed] -l line`8759 UN EX `cu [-dht] [-o | -e ] systemname`8760 **DESCRIPTION**8761 The *cu* utility calls up another system. It manages an interactive conversation, with possible  
8762 transfers of text files.8763 On systems where there are no available communications means (either temporarily or  
8764 permanently), this utility will write an error message describing the problem and exit with a  
8765 non-zero exit status.8766 **OPTIONS**8767 The *cu* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
8768 following options are supported:8769 **-s *speed***

8770 Specify the transmission speed. The default value is device-specific.

8771 **-l *line*** Specify a device name to use as the communication line. This can be used to override  
8772 the search that would otherwise take place for the first available line having the right  
8773 speed. When the **-l** option is used without the **-s** option, the speed of a line is taken  
8774 from the devices file. When the **-l** and **-s** options are both used together, *cu* will search  
8775 the devices file to check if the requested speed for the requested line is available. If so,  
8776 the connection will be made at the requested speed; otherwise, an error message will be  
8777 written and the call will not be made. If the specified device is associated with an  
8778 autodialler, a telephone number must be provided.8779 **-h** Emulate local echo, supporting calls to other computer systems that expect terminals to  
8780 be set to half-duplex mode.8781 **-t** Dial a remote modem that has been set to auto-answer. Appropriate mapping of  
8782 carriage-return to carriage-return-line-feed pairs is set.8783 **-d** Write diagnostic traces.8784 **-o** Designate that odd parity is to be generated for data sent to the remote system.8785 **-e** Designate that even parity is to be generated for data sent to the remote system.8786 **-n** Prompt the user to provide the telephone number to be dialled rather than taking it  
8787 from the command line. This is for added security.8788 **OPERANDS**

8789 The following operands are supported:

8790 ***telno*** When using an automatic dialler, specifies the telephone number with equal signs for  
8791 secondary dial tone or minus signs placed appropriately for delays of 4 seconds.8792 ***systemname***8793 Specifies a *uucp* system name, which can be used rather than a telephone number; in  
8794 this case, *cu* will obtain an appropriate direct line or telephone number from a system  
8795 file.

8796 **STDIN**

8797 The standard input is used to accept data to write to the remote system; see the EXTENDED  
8798 DESCRIPTION section.

8799 **INPUT FILES**

8800 None.

8801 **ENVIRONMENT VARIABLES**

8802 The following environment variables affect the execution of *cu*:

8803 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
8804 *LANG* is unset or null, the corresponding value from the implementation-dependent  
8805 default locale will be used. If any of the internationalisation variables contains an  
8806 invalid setting, the utility will behave as if none of the variables had been defined.

8807 *LC\_ALL*

8808 If set to a non-empty string value, override the values of all the other  
8809 internationalisation variables.

8810 *LC\_CTYPE*

8811 Determine the locale for the interpretation of sequences of bytes of text data as  
8812 characters (for example, single- as opposed to multi-byte characters in arguments and  
8813 input data).

8814 *LC\_MESSAGES*

8815 Determine the locale that should be used to affect the format and contents of diagnostic  
8816 messages written to standard error.

8817 *NLSPATH*

8818 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

8819 **ASYNCHRONOUS EVENTS**

8820 The *cu* utility takes the default action upon receipt of signals, with the exception of:

8821 **SIGHUP**

8822 Close the connection and terminate.

8823 **SIGINT** Forward to the remote system.

8824 **SIGQUIT**

8825 Forward to the remote system.

8826 **SIGUSR1**

8827 Terminate the *cu* process without the normal connection closing sequence.

8828 **STDOUT**

8829 The standard output is used to display data to read from the remote system; see the EXTENDED  
8830 DESCRIPTION section.

8831 **STDERR**

8832 Used only for diagnostic messages.

8833 **OUTPUT FILES**

8834 None.

8835 **EXTENDED DESCRIPTION**

8836 After making the connection, *cu* runs as two processes: the *transmit* process reads data from the  
8837 standard input and, except for lines beginning with "~", passes it to the remote system; the *receive*  
8838 process accepts data from the remote system and, except for lines beginning with "~", passes it to  
8839 the standard output. Normally, an automatic DC3/DC1 protocol is used to control input from

8840 the remote system so that the buffer is not overrun. Lines beginning with "~" have special  
 8841 meanings.

8842 The *transmit* process interprets the following user-initiated commands as:

8843 ~. Terminate the conversation.

8844 ~! Escape to an interactive command interpreter on the local system.

8845 ~!command  
 8846 Execute the shell *command* on the local system.

8847 ~\$command  
 8848 Run the shell *command* locally and send its output to the remote system for execution.

8849 ~%cd Change the directory on the local system.

8850 ~%take *from* [ *to* ]  
 8851 Copy file *from* (on the remote system) to file *to* on the local system. If *to* is omitted, the  
 8852 *from* argument is used in both places.

8853 ~%put *from* [ *to* ]  
 8854 Copy file *from* (on local system) to file *to* on remote system. If *to* is omitted, the *from*  
 8855 argument is used in both places.

8856 ~~ *line* Send the line ~ *line* to the remote system.

8857 ~%break or ~%b  
 8858 Transmit a BREAK to the remote system.

8859 ~%nostop  
 8860 Toggle between DC3/DC1 input control protocol and no input control. This is useful  
 8861 in case the remote system is one that does not respond properly to the DC3 and DC1  
 8862 characters.

8863 The *receive* process normally copies data from the remote system to its standard output.

8864 The use of ~%put requires *stty* and *cat* on the remote side. It also requires that the current erase  
 8865 and kill characters on the remote system be identical to these current control characters on the  
 8866 local system. Backslashes are inserted at appropriate places.

8867 The use of ~%take requires the existence of *echo* and *cat* on the remote system. Also, tabs mode  
 8868 (see *stty*) should be set on the remote system if tabs are to be copied without expansion to  
 8869 spaces.

8870 When *cu* is used on system *X* to connect to system *Y* and subsequently used on system *Y* to  
 8871 connect to system *Z*, commands on system *Y* can be executed by using "~". For example, *uname*  
 8872 can be executed on *Z*, *X* and *Y* as follows:

8873 \$ uname  
 8874 Z  
 8875 \$ ~[X]!uname  
 8876 X  
 8877 \$ ~~[Y]!uname  
 8878 Y

8879 (The darker type indicates system-generated text. The bracketed system names are written by  
 8880 the system after it has recognised the ~! pair, but before it echoed the "!"). In general, "~" causes  
 8881 the command to be executed on the original machine; "~" causes the command to be executed on  
 8882 the next machine in the chain.

8883 **EXIT STATUS**

8884       The following exit values are returned:

8885           0   Successful completion.

8886           &gt;1   An error occurred.

8887 **CONSEQUENCES OF ERRORS**

8888       Default.

8889 **APPLICATION USAGE**

8890       Typical implementations of this utility require a communications line configured to use the **XBD**  
 8891       specification, **Chapter 9, General Terminal Interface**, but other communications means may be  
 8892       used.

8893 **EXAMPLES**

8894           1. To dial a system whose telephone number is 9 1 201 555 1212 using a device-specific speed  
 8895           (where dial tone is expected after the 9):

8896                cu 9=12015551212

8897           2. To login to a system connected by a direct line:

8898                cu -l /dev/ttyXX

8899           or:

8900                cu -l ttyXX

8901           3. To dial a system with the specific line and a specific speed:

8902                cu -s 1200 -l ttyXX

8903           4. To dial a system using a specific line associated with an autodialler:

8904                cu -l culXX 9=12015551212

8905           5. To use a system name:

8906                cu systemname

8907 **FUTURE DIRECTIONS**

8908       None.

8909 **SEE ALSO**8910       *cat, echo, stty, uname, uucp.*8911 **CHANGE HISTORY**

8912       First released in Issue 2.

8913 **Issue 4**

8914       Format reorganised.

8915       Utility Syntax Guidelines support mandated.

8916       Internationalised environment variable support mandated.

8917       Presence of the utility mandated, even on systems where no communications are available.

8918 **Issue 5**

8919       Marked LEGACY.

8920 **NAME**

8921       cut — cut out selected fields of each line of a file

8922 **SYNOPSIS**8923       cut -b *list* [-n] [*file* ...]8924       cut -c *list* [*file* ...]8925       cut -f *list* [-d *delim*][-s][*file* ...]8926 **DESCRIPTION**

8927       The *cut* utility will cut out bytes (**-b** option), characters (**-c** option) or character-delimited fields  
 8928       (**-f** option) from each line in one or more files, concatenate them and write them to standard  
 8929       output.

8930 **OPTIONS**8931       The *cut* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

8932       The option-argument *list* (see options **-b**, **-c** and **-f** below) must be a comma-separated list or  
 8933       blank-character-separated list of positive numbers and ranges. Ranges can be in three forms.  
 8934       The first is two positive numbers separated by a hyphen (*low-high*), which represents all fields  
 8935       from the first number to the second number. The second is a positive number preceded by a  
 8936       hyphen (*-high*), which represents all fields from field number 1 to that number. The third is a  
 8937       positive number followed by a hyphen (*low-*), which represents that number to the last field,  
 8938       inclusive. The elements in *list* can be repeated, can overlap and can be specified in any order.

8939       The following options are supported:

8940       **-b list**   Cut based on a *list* of bytes. Each selected byte will be output unless the **-n** option is  
 8941       also specified. It is not an error to select bytes not present in the input line.

8942       **-c list**   Cut based on a *list* of characters. Each selected character is output. It is not an error to  
 8943       select characters not present in the input line.

8944       **-d delim**  
 8945       Set the field delimiter to the character *delim*. The default is the tab character.

8946       **-f list**   Cut based on a *list* of fields, assumed to be separated in the file by a delimiter character  
 8947       (see **-d**). Each selected field will be output. Output fields will be separated by a single  
 8948       occurrence of the field delimiter character. Lines with no field delimiters will be passed  
 8949       through intact, unless **-s** is specified. It is not an error to select fields not present in the  
 8950       input line.

8951       **-n**       Do not split characters. When specified with the **-b** option, each element in *list* of the  
 8952       form *low-high* (hyphen-separated numbers) will be modified as follows:

8953       — If the byte selected by *low* is not the first byte of a character, *low* will be decremented  
 8954       to select the first byte of the character originally selected by *low*. If the byte selected  
 8955       by *high* is not the last byte of a character, *high* will be decremented to select the last  
 8956       byte of the character prior to the character originally selected by *high*, or zero if there  
 8957       is no prior character. If the resulting range element has *high* equal to zero or *low*  
 8958       greater than *high*, the list element will be dropped from *list* for that input line  
 8959       without causing an error.

8960       Each element in *list* of the form *low-* will be treated as above with *high* set to the  
 8961       number of bytes in the current line, not including the terminating newline character.  
 8962       Each element in *list* of the form *-high* will be treated as above with *low* set to 1. Each  
 8963       element in *list* of the form *num* (a single number) will be treated as above with *low* set to  
 8964       *num* and *high* set to *num*.

8965            **-s**            Suppress lines with no delimiter characters, when used with the **-f** option. Unless  
 8966                            specified, lines with no delimiters will be passed through untouched.

#### 8967 **OPERANDS**

8968            The following operands are supported:

8969            **file**            A pathname of an input file. If no *file* operands are specified, or if a *file* operand is "-",  
 8970                            the standard input will be used.

#### 8971 **STDIN**

8972            The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".  
 8973            See the INPUT FILES section.

#### 8974 **INPUT FILES**

8975            The input files must be text files, except that line lengths are unlimited.

#### 8976 **ENVIRONMENT VARIABLES**

8977            The following environment variables affect the execution of *cut*:

8978            **LANG**            Provide a default value for the internationalisation variables that are unset or null. If  
 8979                            *LANG* is unset or null, the corresponding value from the implementation-dependent  
 8980                            default locale will be used. If any of the internationalisation variables contains an  
 8981                            invalid setting, the utility will behave as if none of the variables had been defined.

8982            **LC\_ALL**

8983                            If set to a non-empty string value, override the values of all the other  
 8984                            internationalisation variables.

8985            **LC\_CTYPE**

8986                            Determine the locale for the interpretation of sequences of bytes of text data as  
 8987                            characters (for example, single- as opposed to multi-byte characters in arguments and  
 8988                            input files).

8989            **LC\_MESSAGES**

8990                            Determine the locale that should be used to affect the format and contents of diagnostic  
 8991                            messages written to standard error.

8992 **EX**            **NLSPATH**

8993                            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

#### 8994 **ASYNCHRONOUS EVENTS**

8995            Default.

#### 8996 **STDOUT**

8997            The *cut* utility output will be a concatenation of the selected bytes, characters or fields (one of the  
 8998            following):

8999                            "%s\n", <concatenation of bytes>

9000                            "%s\n", <concatenation of characters>

9001                            "%s\n", <concatenation of fields and field delimiters>

#### 9002 **STDERR**

9003            Used only for diagnostic messages.

#### 9004 **OUTPUT FILES**

9005            None.

#### 9006 **EXTENDED DESCRIPTION**

9007            None.

9008 **EXIT STATUS**

9009 The following exit values are returned:

9010 0 All input files were output successfully.

9011 &gt;0 An error occurred.

9012 **CONSEQUENCES OF ERRORS**

9013 Default.

9014 **APPLICATION USAGE**

9015 Earlier versions of the *cut* utility worked in an environment where bytes and characters were  
 9016 considered equivalent (modulo backspace and tab character processing in some  
 9017 implementations). In the extended world of multi-byte characters, the new **-b** option has been  
 9018 added. The **-n** option (used with **-b**) allows it to be used to act on bytes rounded to character  
 9019 boundaries. The algorithm specified for **-n** guarantees that:

9020 `cut -b 1-500 -n file > file1`9021 `cut -b 501- -n file > file2`

9022 will end up with all the characters in **file** appearing exactly once in **file1** or **file2**. (There is,  
 9023 however, a newline character in both **file1** and **file2** for each newline character in **file**.)

9024 **EXAMPLES**

9025 Examples of the option qualifier list:

9026 **1,4,7** Select the first, fourth and seventh bytes, characters, or fields and field delimiters.9027 **1-3,8** Equivalent to 1,2,3,8.9028 **-5,10** Equivalent to 1,2,3,4,5,10.9029 **3-** Equivalent to third to last, inclusive.

9030 The *low-high* forms are not always equivalent when used with **-b** and **-n** and multi-byte  
 9031 characters. See the description of **-n**.

9032 The following command:

9033 `cut -d : -f 1,6 /etc/passwd`

9034 reads the System V password file (user database) and produces lines of the form:

9035 `<user ID>:<home directory>`

9036 Most utilities in this specification work on text files. The *cut* utility can be used to turn files with  
 9037 arbitrary line lengths into a set of text files containing the same data. The *paste* utility can be  
 9038 used to create (or recreate) files with arbitrary line lengths. For example, if **file** contains long  
 9039 lines:

9040 `cut -b 1-500 -n file > file1`9041 `cut -b 501- -n file > file2`

9042 creates **file1** (a text file) with lines no longer than 500 bytes (plus the newline character and **file2**  
 9043 that contains the remainder of the data from **file**. (Note that **file2** will not be a text file if there  
 9044 are lines in **file** that are longer than 500 + {LINE\_MAX} bytes.) The original file can be recreated  
 9045 from **file1** and **file2** using the command:

9046 `paste -d "\0" file1 file2 > file`9047 **FUTURE DIRECTIONS**

9048 None.

9049 **SEE ALSO**9050 *grep, paste*, Section 2.5 on page 27.9051 **CHANGE HISTORY**

9052 First released in Issue 2.

9053 **Issue 4**

9054 Aligned with the ISO/IEC 9945-2: 1993 standard.



9055 **NAME**9056 cxref — generate a C-language program cross-reference table (**DEVELOPMENT**)9057 **SYNOPSIS**

```
9058 EX cxref [-cs][-o file][-w num] [-D name[=def]]...[-I dir]...[-U name]...
9059 file ...
```

9060 **DESCRIPTION**

9061 The *cxref* utility analyses a collection of C-language *files* and attempts to build a cross-reference  
 9062 table. Information from **#define** lines is included in the symbol table. A sorted listing is written  
 9063 to standard output of all symbols (auto, static and global) in each *file* separately, or with the **-c**  
 9064 option, in combination. Each symbol contains an asterisk before the declaring reference.

9065 **OPTIONS**

9066 The *cxref* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 9067 that the order of the **-D**, **-I** and **-U** options (which are identical to their interpretation by *c89*) is  
 9068 significant. The following options are supported:

- 9069 **-c** Write a combined cross-reference of all input files.
- 9070 **-w num**  
 9071 Format output no wider than *num* (decimal) columns. This option defaults to 80 if *num*  
 9072 is not specified or is less than 51.
- 9073 **-o file** Direct output to named *file*.
- 9074 **-s** Operate silently; do not print input filenames.

9075 **OPERANDS**

9076 The following operand is supported:

- 9077 *file* A pathname of a C-language source file.

9078 **STDIN**

9079 Not used.

9080 **INPUT FILES**

9081 The input files are C-language source files.

9082 **ENVIRONMENT VARIABLES**

9083 The following environment variables affect the execution of *cxref*:

- 9084 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 9085 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 9086 default locale will be used. If any of the internationalisation variables contains an  
 9087 invalid setting, the utility will behave as if none of the variables had been defined.

9088 **LC\_ALL**

9089 If set to a non-empty string value, override the values of all the other  
 9090 internationalisation variables.

9091 **LC\_COLLATE**

9092 Determine the locale for the ordering of the output.

9093 **LC\_CTYPE**

9094 Determine the locale for the interpretation of sequences of bytes of text data as  
 9095 characters (for example, single- as opposed to multi-byte characters in arguments and  
 9096 input files).

|      |                                                                                                            |
|------|------------------------------------------------------------------------------------------------------------|
| 9097 | <i>LC_MESSAGES</i>                                                                                         |
| 9098 | Determine the locale that should be used to affect the format and contents of diagnostic                   |
| 9099 | messages written to standard error.                                                                        |
| 9100 | <i>NLSPATH</i>                                                                                             |
| 9101 | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                    |
| 9102 | <b>ASYNCHRONOUS EVENTS</b>                                                                                 |
| 9103 | Default.                                                                                                   |
| 9104 | <b>STDOUT</b>                                                                                              |
| 9105 | The standard output is used for the cross-reference listing, unless the <b>-o</b> option is used to select |
| 9106 | a different output file.                                                                                   |
| 9107 | The format of standard output is unspecified, except that the following information is included:           |
| 9108 | • If the <b>-c</b> option is not specified, each portion of the listing starts with the name of the input  |
| 9109 | file on a separate line.                                                                                   |
| 9110 | • The name line is followed by a sorted list of symbols, each with its associated location                 |
| 9111 | pathname, the name of the function in which it appears (if it is not a function name itself),              |
| 9112 | and line number references.                                                                                |
| 9113 | • Each line number may be preceded by an asterisk (*) flag, meaning that this is the declaring             |
| 9114 | reference. Other single-character flags, with implementation-dependent meanings, may be                    |
| 9115 | included.                                                                                                  |
| 9116 | <b>STDERR</b>                                                                                              |
| 9117 | Used only for diagnostic messages.                                                                         |
| 9118 | <b>OUTPUT FILES</b>                                                                                        |
| 9119 | The output file named by the <b>-o</b> option is used instead of standard output.                          |
| 9120 | <b>EXTENDED DESCRIPTION</b>                                                                                |
| 9121 | None.                                                                                                      |
| 9122 | <b>EXIT STATUS</b>                                                                                         |
| 9123 | The following exit values are returned:                                                                    |
| 9124 | 0 Successful completion.                                                                                   |
| 9125 | >0 An error occurred.                                                                                      |
| 9126 | <b>CONSEQUENCES OF ERRORS</b>                                                                              |
| 9127 | Default.                                                                                                   |
| 9128 | <b>APPLICATION USAGE</b>                                                                                   |
| 9129 | None.                                                                                                      |
| 9130 | <b>EXAMPLES</b>                                                                                            |
| 9131 | None.                                                                                                      |
| 9132 | <b>FUTURE DIRECTIONS</b>                                                                                   |
| 9133 | None.                                                                                                      |
| 9134 | <b>SEE ALSO</b>                                                                                            |
| 9135 | <i>cc</i> , <i>c89</i> .                                                                                   |
| 9136 | <b>CHANGE HISTORY</b>                                                                                      |
| 9137 | First released in Issue 2.                                                                                 |

9138 **Issue 4**

9139           Format reorganised.

9140           Utility Syntax Guidelines support mandated.

9141           Internationalised environment variable support mandated.

9142 **Issue 5**

9143           In the SYNOPSIS, [-U dir] is changed to [-U name].

|  
|

9144 **NAME**

9145           date — write the date and time

9146 **SYNOPSIS**9147           date [-u] [+*format*]9148 EX       date [-u] *mmddhhmm*[[*cc*]*yy*]9149 **DESCRIPTION**

9150 EX       The *date* utility writes the date and time to standard output or attempts to set the system date  
 9151           and time. By default, the current date and time will be written. If an operand beginning with "+"  
 9152           is specified, the output format of *date* will be controlled by the field descriptors and other text in  
 9153           the operand.

9154 **OPTIONS**9155           The *date* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

9156           The following option is supported:

9157        -u       Perform operations as if the *TZ* environment variable was set to the string UTC0, or its  
 9158           equivalent historical value of GMT0. Otherwise, *date* will use the timezone indicated  
 9159           by the *TZ* environment variable or the system default if that variable is not set.

9160 **OPERANDS**

9161           The following operands are supported:

9162        +*format*

9163           When the format is specified, each field descriptor will be replaced in the standard  
 9164           output by its corresponding value. All other characters will be copied to the output  
 9165           without change. The output will always be terminated with a newline character.

9166           **Field Descriptors**

9167        %a       Locale's abbreviated weekday name.

9168        %A       Locale's full weekday name.

9169        %b       Locale's abbreviated month name.

9170        %B       Locale's full month name.

9171        %c       Locale's appropriate date and time representation.

9172        %C       Century (a year divided by 100 and truncated to an integer) as a decimal  
 9173           number [00–99].

9174        %d       Day of the month as a decimal number [01–31].

9175        %D       Date in the format *mm/dd/yy*.

9176        %e       Day of the month as a decimal number [1–31] in a two-digit field with leading  
 9177           space character fill.

9178        %h       A synonym for %b.

9179        %H       Hour (24-hour clock) as a decimal number [00–23].

9180        %I       Hour (12-hour clock) as a decimal number [01–12].

9181        %j       Day of the year as a decimal number [001–366].

9182        %m       Month as a decimal number [01–12].

|      |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
|------|-----------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9183 | <b>%M</b>                                                                                                                   | Minute as a decimal number [00–59].                                                                                                                                                                                                                                                                      |
| 9184 | <b>%n</b>                                                                                                                   | A newline character.                                                                                                                                                                                                                                                                                     |
| 9185 | <b>%p</b>                                                                                                                   | Locale's equivalent of either AM or PM.                                                                                                                                                                                                                                                                  |
| 9186 | <b>%r</b>                                                                                                                   | 12-hour clock time [01–12] using the AM/PM notation; in the POSIX locale, this will be equivalent to "%I:%M:%S %p".                                                                                                                                                                                      |
| 9187 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9188 | <b>%S</b>                                                                                                                   | Seconds as a decimal number [00–61].                                                                                                                                                                                                                                                                     |
| 9189 | <b>%t</b>                                                                                                                   | A tab character.                                                                                                                                                                                                                                                                                         |
| 9190 | <b>%T</b>                                                                                                                   | 24-hour clock time [00–23] in the format <i>HH:MM:SS</i> .                                                                                                                                                                                                                                               |
| 9191 | <b>%u</b>                                                                                                                   | Weekday as a decimal number [1 (Monday)–7].                                                                                                                                                                                                                                                              |
| 9192 | <b>%U</b>                                                                                                                   | Week of the year (Sunday as the first day of the week) as a decimal number [00–53]. All days in a new year preceding the first Sunday are considered to be in week 0.                                                                                                                                    |
| 9193 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9194 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9195 | <b>%V</b>                                                                                                                   | Week of the year (Monday as the first day of the week) as a decimal number [01–53]. If the week containing January 1 has four or more days in the new year, then it is considered week 1; otherwise, it is week 53 of the previous year, and the next week is week 1. (See the ISO 8601: 1988 standard.) |
| 9196 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9197 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9198 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9199 | <b>%w</b>                                                                                                                   | Weekday as a decimal number [0 (Sunday)–6].                                                                                                                                                                                                                                                              |
| 9200 | <b>%W</b>                                                                                                                   | Week of the year (Monday as the first day of the week) as a decimal number [00–53]. All days in a new year preceding the first Monday are considered to be in week 0.                                                                                                                                    |
| 9201 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9202 |                                                                                                                             |                                                                                                                                                                                                                                                                                                          |
| 9203 | <b>%x</b>                                                                                                                   | Locale's appropriate date representation.                                                                                                                                                                                                                                                                |
| 9204 | <b>%X</b>                                                                                                                   | Locale's appropriate time representation.                                                                                                                                                                                                                                                                |
| 9205 | <b>%y</b>                                                                                                                   | Year within century [00–99].                                                                                                                                                                                                                                                                             |
| 9206 | <b>%Y</b>                                                                                                                   | Year with century as a decimal number.                                                                                                                                                                                                                                                                   |
| 9207 | <b>%Z</b>                                                                                                                   | Timezone name, or no characters if no timezone is determinable.                                                                                                                                                                                                                                          |
| 9208 | <b>%%</b>                                                                                                                   | A percent sign character.                                                                                                                                                                                                                                                                                |
| 9209 | See the LC_TIME description in the <b>XBD</b> specification, <b>Chapter 5, Locale</b> for the field                         |                                                                                                                                                                                                                                                                                                          |
| 9210 | descriptor values in the POSIX locale.                                                                                      |                                                                                                                                                                                                                                                                                                          |
| 9211 | <b>Modified Field Descriptors</b>                                                                                           |                                                                                                                                                                                                                                                                                                          |
| 9212 | Some field descriptors can be modified by the E and O modifier characters to indicate a                                     |                                                                                                                                                                                                                                                                                                          |
| 9213 | different format or specification as specified in the LC_TIME locale description (see the                                   |                                                                                                                                                                                                                                                                                                          |
| 9214 | <b>XBD</b> specification, <b>Chapter 5, Locale</b> ). If the corresponding keyword (see <b>era</b> ,                        |                                                                                                                                                                                                                                                                                                          |
| 9215 | <b>era_year</b> , <b>era_d_fmt</b> and <b>alt_digits</b> in the <b>XBD</b> specification, <b>Chapter 5, Locale</b> ) is not |                                                                                                                                                                                                                                                                                                          |
| 9216 | specified or not supported for the current locale, the unmodified field descriptor value                                    |                                                                                                                                                                                                                                                                                                          |
| 9217 | will be used.                                                                                                               |                                                                                                                                                                                                                                                                                                          |
| 9218 | <b>%Ec</b>                                                                                                                  | Locale's alternative appropriate date and time representation.                                                                                                                                                                                                                                           |
| 9219 | <b>%EC</b>                                                                                                                  | The name of the base year (period) in the locale's alternative representation.                                                                                                                                                                                                                           |
| 9220 | <b>%Ex</b>                                                                                                                  | Locale's alternative date representation.                                                                                                                                                                                                                                                                |
| 9221 | <b>%EX</b>                                                                                                                  | Locale's alternative time representation.                                                                                                                                                                                                                                                                |

|      |            |                                                                                                                                                   |
|------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 9222 | <b>%Ey</b> | Offset from <b>%EC</b> (year only) in the locale's alternative representation.                                                                    |
| 9223 | <b>%EY</b> | Full alternative year representation.                                                                                                             |
| 9224 | <b>%Od</b> | Day of month using the locale's alternative numeric symbols.                                                                                      |
| 9225 | <b>%Oe</b> | Day of month using the locale's alternative numeric symbols.                                                                                      |
| 9226 | <b>%OH</b> | Hour (24-hour clock) using the locale's alternative numeric symbols.                                                                              |
| 9227 | <b>%OI</b> | Hour (12-hour clock) using the locale's alternative numeric symbols.                                                                              |
| 9228 | <b>%Om</b> | Month using the locale's alternative numeric symbols.                                                                                             |
| 9229 | <b>%OM</b> | Minutes using the locale's alternative numeric symbols.                                                                                           |
| 9230 | <b>%OS</b> | Seconds using the locale's alternative numeric symbols.                                                                                           |
| 9231 | <b>%Ou</b> | Weekday as a number in the locale's alternative representation (Monday = 1).                                                                      |
| 9232 | <b>%OU</b> | Week number of the year (Sunday as the first day of the week) using the locale's alternative numeric symbols.                                     |
| 9233 |            |                                                                                                                                                   |
| 9234 | <b>%OV</b> | Week number of the year (Monday as the first day of the week, rules corresponding to <b>%V</b> ), using the locale's alternative numeric symbols. |
| 9235 |            |                                                                                                                                                   |
| 9236 | <b>%Ow</b> | Weekday as a number in the locale's alternative representation (Sunday = 0).                                                                      |
| 9237 | <b>%OW</b> | Week number of the year (Monday as the first day of the week) using the locale's alternative numeric symbols.                                     |
| 9238 |            |                                                                                                                                                   |
| 9239 | <b>%Oy</b> | Year (offset from <b>%C</b> ) in alternative representation.                                                                                      |

|      |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9240 | EX | <b>mmddhhmm</b> [ [ <i>cc</i> ] <i>yy</i> ]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 9241 |    | Attempt to set the system date and time from the value given in the operand. This is only possible if the user has appropriate privileges and the system permits the setting of the system date and time. The first <i>mm</i> is the month (number); <i>dd</i> is the day (number); <i>hh</i> is the hour (number, 24-hour system); the second <i>mm</i> is the minute (number); <i>cc</i> is the century and is the first two digits of the year (this is optional); <i>yy</i> is the last two digits of the year and is optional. If century is not specified, then values in the range [69-99] refer to years in the twentieth century (1969 to 1999 inclusive), and values in the range [00-68] refer to years in the twenty-first century (2000 to 2068 inclusive). |
| 9242 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9243 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9244 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9245 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9246 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9247 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9248 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 9249 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

## 9250 STDIN

9251 Not used.

## 9252 INPUT FILES

9253 None.

## 9254 ENVIRONMENT VARIABLES

9255 The following environment variables affect the execution of *date*:

9256 **LANG** Provide a default value for the internationalisation variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

9260 **LC\_ALL**

9261 If set to a non-empty string value, override the values of all the other internationalisation variables.

9262

9263 **LC\_CTYPE**  
 9264 Determine the locale for the interpretation of sequences of bytes of text data as  
 9265 characters (for example, single- as opposed to multi-byte characters in arguments).

9266 **LC\_MESSAGES**  
 9267 Determine the locale that should be used to affect the format and contents of diagnostic  
 9268 messages written to standard error.

9269 **LC\_TIME**  
 9270 Determine the format and contents of date and time strings written by *date*.

9271 EX **NLSPATH**  
 9272 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

9273 **TZ** Determine the timezone in which the time and date are written, unless the **-u** option is  
 9274 specified. If the *TZ* variable is not set and the **-u** is not specified, an unspecified system  
 9275 default timezone is used.

9276 **ASYNCHRONOUS EVENTS**  
 9277 Default.

9278 **STDOUT**  
 9279 When no formatting operand is specified, the output in the POSIX locale is equivalent to  
 9280 specifying:  
 9281 `date "+%a %b %e %H:%M:%S %Z %Y"`

9282 **STDERR**  
 9283 Used only for diagnostic messages.

9284 **OUTPUT FILES**  
 9285 None.

9286 **EXTENDED DESCRIPTION**  
 9287 None.

9288 **EXIT STATUS**  
 9289 The following exit values are returned:  
 9290 0 The date was written successfully.  
 9291 >0 An error occurred.

9292 **CONSEQUENCES OF ERRORS**  
 9293 Default.

9294 **APPLICATION USAGE**  
 9295 Field descriptors are of unspecified format when not in the POSIX locale. Some of them can  
 9296 contain newline characters in some locales, so it may be difficult to use the format shown in  
 9297 standard output for parsing the output of *date* in those locales.  
 9298 The range of values for **%S** extends from 0 to 61 seconds to accommodate the occasional leap  
 9299 second or double leap second.  
 9300 Although certain of the field descriptors in the POSIX locale (such as the name of the month) are  
 9301 shown with initial capital letters, this need not be the case in other locales. Programs using these  
 9302 fields may need to adjust the capitalisation if the output is going to be used at the beginning of a  
 9303 sentence.  
 9304 The date string formatting capabilities are intended for use in Gregorian style calendars,  
 9305 possibly with a different starting year (or years). The **%x** and **%c** field descriptors, however, are  
 9306 intended for local representation; these may be based on a different, non-Gregorian calendar.

The %C field descriptor was introduced to allow a fallback for the %EC (alternative year format base year); it can be viewed as the base of the current subdivision in the Gregorian calendar. A century is not calculated as an ordinal number; this Guide was published in century 19, not the twentieth. Both the %Ey and %y can then be viewed as the offset from %EC and %C, respectively.

The E and O modifiers modify the traditional field descriptors, so that they can always be used, even if the implementation (or the current locale) does not support the modifier.

The E modifier supports alternative date formats, such as the Japanese Emperor's Era, as long as these are based on the Gregorian calendar system. Extending the E modifiers to other date elements may provide an implementation-dependent extension capable of supporting other calendar systems, especially in combination with the O modifier.

The O modifier supports time and date formats using the locale's alternative numerical symbols, such as Kanji or Hindi digits or ordinal number representation.

Non-European locales, whether they use Latin digits in computational items or not, often have local forms of the digits for use in date formats. This is not totally unknown even in Europe; a variant of dates uses Roman numerals for the months: the third day of September 1991 would be written as 3.IX.1991. In Japan, Kanji digits are regularly used for dates; in Arabic-speaking countries, Hindi digits are used. The %d, %e, %H, %I, %m, %S, %U, %w, %W and %y field descriptors always return the date and time field in Latin digits (that is, 0 to 9). The %O modifier was introduced to support the use for display purposes of non-Latin digits. In the LC\_TIME category in *localedef*, the optional **alt\_digits** keyword is intended for this purpose. As an example, assume the following (partial) *localedef* source:

```
alt_digits " ";"I ";"II ";"III ";"IV ";"V ";"VI ";"VII ";"VIII" \
 "IX ";"X ";"XI ";"XII"
d_fmt "%e.%Om.%Y"
```

With the above date, the command:

```
date "+%x"
```

would yield 3.IX.1991. With the same **d\_fmt**, but without the **alt\_digits**, the command would yield 3.9.1991.

## EXAMPLES

1. The following are input/output examples of *date* used at arbitrary times in the POSIX locale:

```
$ date
Tue Jun 26 09:58:10 PDT 1990

$ date "+DATE: %m/%d/%y%nTIME: %H:%M:%S"
DATE: 11/02/91
TIME: 13:36:16

$ date "+TIME: %r"
TIME: 01:36:32 PM
```

2. Examples for Denmark, where the default date and time format is "%a %d %b %Y %T %Z":

```
$ LANG=da_DK.iso_8859-1 date
ons 02 okt 1991 15:03:32 CET
```



```

9350 $ LANG=da_DK.iso_8859-1 date "+DATO: %A den %e. %B nKLOKKEN: %H:%M:%S"
9351 DATO: onsdag den 2. oktober 1991
9352 KLOKKEN: 15:03:56

9353 3. Examples for Germany, where the default date and time format is
9354 "%a %d.%h.%Y, %T %Z":

9355 $ LANG=De_DE.88591 date
9356 Mi 02.Okt.1991, 15:01:21 MEZ

9357 $ LANG=De_DE.88591 date "+DATUM: %A, %d. %B nZEIT: %H:%M:%S"
9358 DATUM: Mittwoch, 02. Oktober 1991
9359 ZEIT: 15:02:02

9360 4. Examples for France, where the default date and time format is "%a %d %h %Y %Z %T":

9361 $ LANG=Fr_FR.88591 date
9362 Mer 02 oct 1991 MET 15:03:32

9363 $ LANG=Fr_FR.88591 date "+JOUR: %A %d %B nHEURE: %H:%M:%S"
9364 JOUR: Mercredi 02 octobre 1991
9365 HEURE: 15:03:56

9366 FUTURE DIRECTIONS
9367 None.

9368 SEE ALSO
9369 The XSH specification description of ctime(), printf().

9370 CHANGE HISTORY
9371 First released in Issue 2.

9372 Issue 4
9373 Aligned with the ISO/IEC 9945-2: 1993 standard.

9374 Issue 5
9375 Changes are made for Year 2000 alignment.

```

## 9376 NAME

9377 dd — convert and copy a file

## 9378 SYNOPSIS

9379 dd [*operand* ...]

## 9380 DESCRIPTION

9381 The *dd* utility will copy the specified input file to the specified output file with possible  
 9382 conversions using specific input and output block sizes. It will read the input one block at a  
 9383 time, using the specified input block size; it then will process the block of data actually returned,  
 9384 which could be smaller than the requested block size. It will apply any conversions that have  
 9385 been specified and write the resulting data to the output in blocks of the specified output block  
 9386 size. If the **bs=expr** operand is specified and no conversions other than **sync**, **noerror** or **notrunc**  
 9387 are requested, the data returned from each input block will be written as a separate output  
 9388 block; if the read returns less than a full block and the **sync** conversion is not specified, the  
 9389 resulting output block will be the same size as the input block. If the **bs=expr** operand is not  
 9390 specified, or a conversion other than **sync**, **noerror** or **notrunc** is requested, the input will be  
 9391 processed and collected into full-sized output blocks until the end of the input is reached.

9392 The processing order is as follows:

- 9393 1. An input block is read.
- 9394 2. If the input block is shorter than the specified input block size and the **sync** conversion is  
 9395 specified, null bytes are appended to the input data up to the specified size. (If either  
 9396 **block** or **unblock** is also specified, space characters are appended instead of null bytes.)  
 9397 The remaining conversions and output include the pad characters as if they had been read  
 9398 from the input.
- 9399 3. If the **bs=expr** operand is specified and no conversion other than **sync** or **noerror** is  
 9400 requested, the resulting data will be written to the output as a single block, and the  
 9401 remaining steps are omitted.
- 9402 4. If the **swab** conversion is specified, each pair of input data bytes will be swapped. If there  
 9403 is an odd number of bytes in the input block, the results are unspecified.
- 9404 5. Any remaining conversions (**block**, **unblock**, **lcase** and **ucase**) will be performed. These  
 9405 conversions will operate on the input data independently of the input blocking; an input or  
 9406 output fixed-length record may span block boundaries.
- 9407 6. The data resulting from input or conversion or both will be aggregated into output blocks  
 9408 of the specified size. After the end of input is reached, any remaining output will be  
 9409 written as a block without padding if **conv=sync** is not specified; thus the final output  
 9410 block may be shorter than the output block size.

## 9411 OPTIONS

9412 None.

## 9413 OPERANDS

9414 The following operands are supported:

- 9415 **if=file** Specify the input pathname; the default is standard input.
- 9416 **of=file** Specify the output pathname; the default is standard output. If the **seek=expr**  
 9417 conversion is not also specified, the output file will be truncated before the copy begins,  
 9418 unless **conv=notrunc** is specified. If **seek=expr** is specified, but **conv=notrunc** is not,  
 9419 the effect of the copy will be to preserve the blocks in the output file over which *dd*  
 9420 seeks, but no other portion of the output file will be preserved. (If the size of the seek  
 9421 plus the size of the input file is less than the previous size of the output file, the output

9422 file will be shortened by the copy.)

9423 **ibs=expr**

9424 Specify the input block size, in bytes, by *expr* (default is 512).

9425 **obs=expr**

9426 Specify the output block size, in bytes, by *expr* (default is 512).

9427 **bs=expr** Set both input and output block sizes to *expr* bytes, superseding **ibs=** and **obs=**. If no

9428 conversion other than **sync**, **noerror** and **notrunc** is specified, each input block will be

9429 copied to the output as a single block without aggregating short blocks.

9430 **cbs=expr**

9431 Specify the conversion block size for **block** and **unblock** in bytes by *expr* (default is

9432 zero). If **cbs=** is omitted or given a value of zero, using **block** or **unblock** produces

9433 unspecified results.

9434 EX This operand must also be specified if the **conv=** operand is specified with a value of

9435 **ascii**, **ebcdic** or **ibm**. For a **conv=** operand with an **ascii** value, the input is handled as

9436 described for the **unblock** value, except that characters are converted to ASCII before

9437 any trailing space characters are deleted. For **conv=** operands with **ebcdic** or **ibm**

9438 values, the input is handled as described for the **block** value except that the characters

9439 are converted to EBCDIC or IBM EBCDIC, respectively, after any trailing space

9440 characters are added.

9441 **skip=n** Skip *n* input blocks (using the specified input block size) before starting to copy. On

9442 seekable files, the implementation will read the blocks or seek past them; on non-

9443 seekable files, the blocks will be read and the data will be discarded.

9444 **seek=n** Skip *n* blocks (using the specified output block size) from beginning of output file

9445 before copying. On non-seekable files, existing blocks will be read and space from the

9446 current end-of-file to the specified offset, if any, filled with null bytes; on seekable files,

9447 the implementation will seek to the specified offset or read the blocks as described for

9448 non-seekable files.

9449 **count=n**

9450 Copy only *n* input blocks.

9451 **conv=value[ , value ... ]**

9452 Where *values* are comma-separated symbols from the following list.

9453 EX **ascii** Convert EBCDIC to ASCII. See Table 3-4 on page 261.

9454 EX **ebcdic** Convert ASCII to EBCDIC. See Table 3-4 on page 261.

9455 EX **ibm** Convert ASCII to a different EBCDIC set. See Table 3-5 on page 262.

9456 The **ascii**, **ebcdic** and **ibm** values are mutually exclusive.

9457 **block** Treat the input as a sequence of newline-character-terminated or end-of-file-

9458 terminated variable-length records independent of the input block

9459 boundaries. Each record is converted to a record with a fixed length specified

9460 by the conversion block size. Any newline character is removed from the

9461 input line; space characters are appended to lines that are shorter than their

9462 conversion block size to fill the block. Lines that are longer than the

9463 conversion block size are truncated to the largest number of characters that

9464 will fit into that size; the number of truncated lines is reported (see the

9465 STDERR section).

|      |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9466 |                | The <b>block</b> and <b>unblock</b> values are mutually exclusive.                                                                                                                                                                                                                                                                                                                                                                        |
| 9467 | <b>unblock</b> | Convert fixed-length records to variable length. Read a number of bytes equal to the conversion block size (or the number of bytes remaining in the input, if less than the conversion block size), delete all trailing space characters, and append a newline character.                                                                                                                                                                 |
| 9468 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9469 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9470 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9471 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9472 | <b>lcase</b>   | Map upper-case characters specified by the LC_CTYPE keyword <b>tolower</b> to the corresponding lower-case character. Characters for which no mapping is specified will not be modified by this conversion.                                                                                                                                                                                                                               |
| 9473 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9474 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9475 |                | The <b>lcase</b> and <b>ucase</b> symbols are mutually exclusive.                                                                                                                                                                                                                                                                                                                                                                         |
| 9476 | <b>ucase</b>   | Map lower-case characters specified by the LC_CTYPE keyword <b>toupper</b> to the corresponding upper-case character. Characters for which no mapping is specified will not be modified by this conversion.                                                                                                                                                                                                                               |
| 9477 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9478 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9479 | <b>swab</b>    | Swap every pair of input bytes. If the current input record is an odd number of bytes, the last byte in the input record is ignored.                                                                                                                                                                                                                                                                                                      |
| 9480 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9481 | <b>noerror</b> | Do not stop processing on an input error. When an input error occurs, a diagnostic message will be written on standard error, followed by the current input and output block counts in the same format as used at completion (see the STDERR section). If the <b>sync</b> conversion is specified, the missing input will be replaced with null bytes and processed normally; otherwise, the input block will be omitted from the output. |
| 9482 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9483 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9484 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9485 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9486 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9487 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9488 | <b>notrunc</b> | Do not truncate the output file. Preserve blocks in the output file not explicitly written by this invocation of the <i>dd</i> utility. (See also the preceding <b>of=file</b> operand.)                                                                                                                                                                                                                                                  |
| 9489 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9490 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9491 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9492 | <b>sync</b>    | Pad every input block to the size of the <b>ibs=</b> buffer, appending null bytes. (If either <b>block</b> or <b>unblock</b> is also specified, append space characters, rather than null bytes.)                                                                                                                                                                                                                                         |
| 9493 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 9494 |                |                                                                                                                                                                                                                                                                                                                                                                                                                                           |

9495 The behaviour is unspecified if operands other than **conv=** are specified more than once.

9496 For the **bs=**, **cbs=**, **ibs=** and **obs=** operands, the application must supply an expression  
 9497 specifying a size in bytes. The expression, *expr*, can be:

- 9498 1. a positive decimal number
- 9499 2. a positive decimal number followed by k, specifying multiplication by 1024
- 9500 3. a positive decimal number followed by b, specifying multiplication by 512
- 9501 4. two or more positive decimal numbers (with or without k or b) separated by x, specifying  
 9502 the product of the indicated values.

9503 All of the operands will be processed before any input is read.

9504 EX The following two tables display the octal number character values used for the **ascii** and **ebcdic**  
 9505 conversions (first table) and for the **ibm** conversion (second table). In both tables, the ASCII  
 9506 values are the row and column headers and the EBCDIC values are found at their intersections.  
 9507 For example, ASCII 0012 (LF) is the second row, third column, yielding 0045 in EBCDIC. The  
 9508 inverted tables (for EBCDIC to ASCII conversion) are not shown, but are in one-to-one  
 9509 correspondence with these tables. The differences between the two tables are highlighted by

9510

9511

small boxes drawn around five entries.

|             | 0        | 1        | 2        | 3        | 4        | 5        | 6        | 7        |
|-------------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>0000</b> | 0000 NUL | 0001 SOH | 0002 STX | 0003 ETX | 0067 EOT | 0055 ENQ | 0056 ACK | 0057 BEL |
| <b>0010</b> | 0026 BS  | 0005 HT  | 0045 LF  | 0013 VT  | 0014 FF  | 0015 CR  | 0016 SO  | 0017 SI  |
| <b>0020</b> | 0020 DLE | 0021 DC1 | 0022 DC2 | 0023 DC3 | 0074 DC4 | 0075 NAK | 0062 SYN | 0046 ETB |
| <b>0030</b> | 0030 CAN | 0031 EM  | 0077 SUB | 0047 ESC | 0034 IFS | 0035 IGS | 0036 IRS | 0037 ITB |
| <b>0040</b> | 0100 Sp  | 0132 !   | 0177 "   | 0173 #   | 0133 \$  | 0154 %   | 0120 &   | 0175 '   |
| <b>0050</b> | 0115 (   | 0135 )   | 0134 *   | 0116 +   | 0153 ,   | 0140 -   | 0113 .   | 0141 /   |
| <b>0060</b> | 0360 0   | 0361 1   | 0362 2   | 0363 3   | 0364 4   | 0365 5   | 0366 6   | 0367 7   |
| <b>0070</b> | 0370 8   | 0371 9   | 0172 :   | 0136 ;   | 0114 <   | 0176 =   | 0156 >   | 0157 ?   |
| <b>0100</b> | 0174 @   | 0301 A   | 0302 B   | 0303 C   | 0304 D   | 0305 E   | 0306 F   | 0307 G   |
| <b>0110</b> | 0310 H   | 0311 I   | 0321 J   | 0322 K   | 0323 L   | 0324 M   | 0325 N   | 0326 O   |
| <b>0120</b> | 0327 P   | 0330 Q   | 0331 R   | 0342 S   | 0343 T   | 0344 U   | 0345 V   | 0346 W   |
| <b>0130</b> | 0347 X   | 0350 Y   | 0351 Z   | 0255 [   | 0340 \   | 0275 ]   | 0232     | 0155 _   |
| <b>0140</b> | 0171 `   | 0201 a   | 0202 b   | 0203 c   | 0204 d   | 0205 e   | 0206 f   | 0207 g   |
| <b>0150</b> | 0210 h   | 0211 i   | 0221 j   | 0222 k   | 0223 l   | 0224 m   | 0225 n   | 0226 o   |
| <b>0160</b> | 0227 p   | 0230 q   | 0231 r   | 0242 s   | 0243 t   | 0244 u   | 0245 v   | 0246 w   |
| <b>0170</b> | 0247 x   | 0250 y   | 0251 z   | 0300 {   | 0117     | 0320 }   | 0137 ~   | 0007 DEL |
| <b>0200</b> | 0040 DS  | 0041 SOS | 0042 FS  | 0043 WUS | 0044 BYP | 0025 NL  | 0006 RNL | 0027 POC |
| <b>0210</b> | 0050 SA  | 0051 SFE | 0052 SM  | 0053 CSP | 0054 MFA | 0011 SPS | 0012 RPT | 0033 CU1 |
| <b>0220</b> | 0060     | 0061     | 0032 UBS | 0063 IR  | 0064 PP  | 0065 TRN | 0066 NBS | 0010 GE  |
| <b>0230</b> | 0070 SBS | 0071 IT  | 0072 RFF | 0073 CU3 | 0004 SEL | 0024 RES | 0076     | 0341     |
| <b>0240</b> | 0101     | 0102     | 0103     | 0104     | 0105     | 0106     | 0107     | 0110     |
| <b>0250</b> | 0111     | 0121     | 0122     | 0123     | 0124     | 0125     | 0126     | 0127     |
| <b>0260</b> | 0130     | 0131     | 0142     | 0143     | 0144     | 0145     | 0146     | 0147     |
| <b>0270</b> | 0150     | 0151     | 0160     | 0161     | 0162     | 0163     | 0164     | 0165     |
| <b>0300</b> | 0166     | 0167     | 0170     | 0200     | 0212     | 0213     | 0214     | 0215     |
| <b>0310</b> | 0216     | 0217     | 0220     | 0152 ¡   | 0233     | 0234     | 0235     | 0236     |
| <b>0320</b> | 0237     | 0240     | 0252     | 0253     | 0254     | 0112 ¢   | 0256     | 0257     |
| <b>0330</b> | 0260     | 0261     | 0262     | 0263     | 0264     | 0265     | 0266     | 0267     |
| <b>0340</b> | 0270     | 0271     | 0272     | 0273     | 0274     | 0241     | 0276     | 0277     |
| <b>0350</b> | 0312     | 0313     | 0314 ¢   | 0315     | 0316 ¥   | 0317     | 0332     | 0333     |
| <b>0360</b> | 0334     | 0335     | 0336     | 0337     | 0352     | 0353     | 0354 ¢   | 0355     |
| <b>0370</b> | 0356     | 0357     | 0372     | 0373     | 0374     | 0375     | 0376     | 0377 EO  |

Table 3-4 ASCII to EBCDIC Conversion

9512

9513

|             | 0        | 1        | 2        | 3        | 4        | 5        | 6        | 7        |
|-------------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>0000</b> | 0000 NUL | 0001 SOH | 0002 STX | 0003 ETX | 0067 EOT | 0055 ENQ | 0056 ACK | 0057 BEL |
| <b>0010</b> | 0026 BS  | 0005 HT  | 0045 LF  | 0013 VT  | 0014 FF  | 0015 CR  | 0016 SO  | 0017 SI  |
| <b>0020</b> | 0020 DLE | 0021 DC1 | 0022 DC2 | 0023 DC3 | 0074 DC4 | 0075 NAK | 0062 SYN | 0046 ETB |
| <b>0030</b> | 0030 CAN | 0031 EM  | 0077 SUB | 0047 ESC | 0034 IFS | 0035 IGS | 0036 IRS | 0037 ITB |
| <b>0040</b> | 0100 Sp  | 0132 !   | 0177 "   | 0173 #   | 0133 \$  | 0154 %   | 0120 &   | 0175 '   |
| <b>0050</b> | 0115 (   | 0135 )   | 0134 *   | 0116 +   | 0153 ,   | 0140 -   | 0113 .   | 0141 /   |
| <b>0060</b> | 0360 0   | 0361 1   | 0362 2   | 0363 3   | 0364 4   | 0365 5   | 0366 6   | 0367 7   |
| <b>0070</b> | 0370 8   | 0371 9   | 0172 :   | 0136 ;   | 0114 <   | 0176 =   | 0156 >   | 0157 ?   |
| <b>0100</b> | 0174 @   | 0301 A   | 0302 B   | 0303 C   | 0304 D   | 0305 E   | 0306 F   | 0307 G   |
| <b>0110</b> | 0310 H   | 0311 I   | 0321 J   | 0322 K   | 0323 L   | 0324 M   | 0325 N   | 0326 O   |
| <b>0120</b> | 0327 P   | 0330 Q   | 0331 R   | 0342 S   | 0343 T   | 0344 U   | 0345 V   | 0346 W   |
| <b>0130</b> | 0347 X   | 0350 Y   | 0351 Z   | 0255 [   | 0340 \   | 0275 ]   | 0137 ¬   | 0155 _   |
| <b>0140</b> | 0171 `   | 0201 a   | 0202 b   | 0203 c   | 0204 d   | 0205 e   | 0206 f   | 0207 g   |
| <b>0150</b> | 0210 h   | 0211 i   | 0221 j   | 0222 k   | 0223 l   | 0224 m   | 0225 n   | 0226 o   |
| <b>0160</b> | 0227 p   | 0230 q   | 0231 r   | 0242 s   | 0243 t   | 0244 u   | 0245 v   | 0246 w   |
| <b>0170</b> | 0247 x   | 0250 y   | 0251 z   | 0300 {   | 0117     | 0320 }   | 0241 ¯   | 0007 DEL |
| <b>0200</b> | 0040 DS  | 0041 SOS | 0042 FS  | 0043 WUS | 0044 BYP | 0025 NL  | 0006 RNL | 0027 POC |
| <b>0210</b> | 0050 SA  | 0051 SFE | 0052 SM  | 0053 CSP | 0054 MFA | 0011 SPS | 0012 RPT | 0033 CU1 |
| <b>0220</b> | 0060     | 0061     | 0032 UBS | 0063 IR  | 0064 PP  | 0065 TRN | 0066 NBS | 0010 GE  |
| <b>0230</b> | 0070 SBS | 0071 IT  | 0072 RFF | 0073 CU3 | 0004 SEL | 0024 RES | 0076     | 0341     |
| <b>0240</b> | 0101     | 0102     | 0103     | 0104     | 0105     | 0106     | 0107     | 0110     |
| <b>0250</b> | 0111     | 0121     | 0122     | 0123     | 0124     | 0125     | 0126     | 0127     |
| <b>0260</b> | 0130     | 0131     | 0142     | 0143     | 0144     | 0145     | 0146     | 0147     |
| <b>0270</b> | 0150     | 0151     | 0160     | 0161     | 0162     | 0163     | 0164     | 0165     |
| <b>0300</b> | 0166     | 0167     | 0170     | 0200     | 0212     | 0213     | 0214     | 0215     |
| <b>0310</b> | 0216     | 0217     | 0220     | 0232     | 0233     | 0234     | 0235     | 0236     |
| <b>0320</b> | 0237     | 0240     | 0252     | 0253     | 0254     | 0255 [   | 0256     | 0257     |
| <b>0330</b> | 0260     | 0261     | 0262     | 0263     | 0264     | 0265     | 0266     | 0267     |
| <b>0340</b> | 0270     | 0271     | 0272     | 0273     | 0274     | 0275 ]   | 0276     | 0277     |
| <b>0350</b> | 0312     | 0313     | 0314 J   | 0315     | 0316 Y   | 0317     | 0332     | 0333     |
| <b>0360</b> | 0334     | 0335     | 0336     | 0337     | 0352     | 0353     | 0354 H   | 0355     |
| <b>0370</b> | 0356     | 0357     | 0372     | 0373     | 0374     | 0375     | 0376     | 0377 EO  |

Table 3-5 ASCII to IBM EBCDIC Conversion

9514

9515 **STDIN**

9516 If no **if=** operand is specified, the standard input will be used. See the INPUT FILES section.

9517 **INPUT FILES**

9518 The input file can be any file type.

9519 **ENVIRONMENT VARIABLES**

9520 The following environment variables affect the execution of *dd*:

9521 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 9522 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 9523 default locale will be used. If any of the internationalisation variables contains an  
 9524 invalid setting, the utility will behave as if none of the variables had been defined.

9525 **LC\_ALL**

9526 If set to a non-empty string value, override the values of all the other  
 9527 internationalisation variables.

9528 **LC\_CTYPE**

9529 Determine the locale for the interpretation of sequences of bytes of text data as  
 9530 characters (for example, single- as opposed to multi-byte characters in arguments and  
 9531 input files), the classification of characters as upper- or lower-case, and the mapping of  
 9532 characters from one case to the other.

9533 **LC\_MESSAGES**

9534 Determine the locale that should be used to affect the format and contents of diagnostic  
 9535 messages written to standard error and informative messages written to standard  
 9536 output.

9537 **EX NLSPATH**

9538 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

9539 **ASYNCHRONOUS EVENTS**

9540 For SIGINT, the *dd* utility will write status information to standard error before exiting. It will  
 9541 take the standard action for all other signals; see the ASYNCHRONOUS EVENTS section in  
 9542 Section 1.9 on page 11.

9543 **STDOUT**

9544 If no **of=** operand is specified, the standard output will be used. The nature of the output  
 9545 depends on the operands selected.

9546 **STDERR**

9547 On completion, *dd* will write the number of input and output blocks to standard error. In the  
 9548 POSIX locale the following formats will be used:

9549 "%u+%u records in\n", <number of whole input blocks>,  
 9550 <number of partial input blocks>

9551 "%u+%u records out\n", <number of whole output blocks>,  
 9552 <number of partial output blocks>

9553 A partial input block is one for which *read()* returned less than the input block size. A partial  
 9554 output block is one that was written with fewer bytes than specified by the output block size.

9555 In addition, when there is at least one truncated block, the number of truncated blocks will be  
 9556 written to standard error. In the POSIX locale, the format is:

9557 "%u truncated %s\n", <number of truncated blocks>, "record" (if  
 9558 <number of truncated blocks> is one) "records" (otherwise)

9559 Diagnostic messages may also be written to standard error.

#### 9560 OUTPUT FILES

9561 If the **of=** operand is used, the output will be the same as described in the STDOUT section. |

#### 9562 EXTENDED DESCRIPTION

9563 None.

#### 9564 EXIT STATUS

9565 The following exit values are returned:

9566 0 The input file was copied successfully.

9567 >0 An error occurred.

#### 9568 CONSEQUENCES OF ERRORS

9569 If an input error is detected and the **noerror** conversion has not been specified, any partial  
9570 output block will be written to the output file, a diagnostic message will be written, and the copy  
9571 operation will be discontinued. If some other error is detected, a diagnostic message will be  
9572 written and the copy operation will be discontinued.

#### 9573 APPLICATION USAGE

9574 The input and output block size can be specified to take advantage of raw physical I/O. |

9575 There are many different versions of the EBCDIC codesets. The ASCII and EBCDIC conversions |  
9576 specified for the *dd* utility perform conversions for the version specified by the tables.

#### 9577 EXAMPLES

9578 The following command:

9579 `dd if=/dev/rmt0h of=/dev/rmt1h`

9580 copies from tape drive 0 to tape drive 1, using a common historical device naming convention.

9581 The following command:

9582 `dd ibs=10 skip=1`

9583 strips the first 10 bytes from standard input.

9584 This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the  
9585 ASCII file **x**:

9586 `dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase`

#### 9587 FUTURE DIRECTIONS

9588 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
9589 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
9590 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
9591 finalised. |

#### 9592 SEE ALSO

9593 *sed*, *tr*.

#### 9594 CHANGE HISTORY

9595 First released in Issue 2. |

#### 9596 Issue 4

9597 Aligned with the ISO/IEC 9945-2: 1993 standard. |



9598 **Issue 5**

9599 The second paragraph of the **cbs=** description is reworded and marked EX.

9600 FUTURE DIRECTIONS section added.

## 9601 NAME

9602       delta — make a delta (change) to an SCCS file (**DEVELOPMENT**)

## 9603 SYNOPSIS

9604 EX       delta [-nps][-g *list*][-m *mrlist*][-r *SID*][-y[*comment*]] *file...*

## 9605 DESCRIPTION

9606       The *delta* utility is used to permanently introduce into the named SCCS files changes that were  
9607       made to the files retrieved by *get* (called the *g-files*, or generated files).

## 9608 OPTIONS

9609       The *delta* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
9610       that the **-y** option has an optional option-argument. This optional option-argument cannot be  
9611       presented as a separate argument. The following options are supported:

9612       **-r** *SID*   Uniquely identify which delta is to be made to the SCCS file. The use of this option is  
9613       necessary only if two or more outstanding *get* commands for editing (*get -e*) on the  
9614       same SCCS file were done by the same person (login name). The *SID* value specified  
9615       with the **-r** option can be either the *SID* specified on the *get* command line or the *SID* to  
9616       be made as reported by the *get* utility; see *get*.

9617       **-s**       Suppress the report to standard output of the activity associated with each *file*. See the  
9618       STDOUT section.

9619       **-n**       Specify retention of the edited *g-file* (normally removed at completion of delta  
9620       processing).

9621       **-g** *list*   Specify a *list*, (see *get* for the definition of *list*) of deltas that are to be ignored when the  
9622       file is accessed at the change level (*SID*) created by this delta.

9623       **-m** *mrlist*

9624       Specify a modification request (MR) number that must be supplied as the reason for  
9625       creating the new delta. This is used if the SCCS file has the **v** flag set; see *admin*.

9626       If **-m** is not used and the standard input is a terminal, the prompt described in the  
9627       STDOUT section is written to standard output before the standard input is read; if the  
9628       standard input is not a terminal, no prompt is issued.

9629       MRs in a list are separated by blanks. An unescaped newline character terminates the  
9630       MR list.

9631       Note that if the **v** flag has a value, it is taken to be the name of a program which will  
9632       validate the correctness of the MR numbers. If a non-zero exit status is returned from  
9633       the MR number validation program, *delta* terminates. (It is assumed that the MR  
9634       numbers were not all valid.)

9635       **-y**[*comment*]

9636       Describe the reason for making the delta. This is an arbitrary group of lines that would  
9637       meet the definition of a text file. Systems support *comments* from zero to 512 bytes and  
9638       may support longer values. A null string (specified as either **-y**, **-y"** or in response to  
9639       a prompt for a comment) is considered a valid *comment*.

9640       If **-y** is not specified and the standard input is a terminal, the prompt described in the  
9641       STDOUT section is written to standard output before the standard input is read; if the  
9642       standard input is not a terminal, no prompt is issued. An unescaped newline character  
9643       terminates the comment text.

9644       The **-y** option is required if the *file* operand is specified as **-**.

9645           **-p**       Write (to standard output) the SCCS file differences before and after the delta is applied  
 9646                      in *diff* format; see *diff*.

## 9647 OPERANDS

9648           The following operands are supported:

9649           *file*       A pathname of an existing SCCS file or a directory. If *file* is a directory, *delta* behaves as  
 9650                      though each file in the directory were specified as a named file, except that non-SCCS  
 9651                      files (last component of the pathname does not begin with s.) and unreadable files are  
 9652                      silently ignored.

9653                      If a single instance *file* is specified as *-*, the standard input is read; each line of the  
 9654                      standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files  
 9655                      and unreadable files are silently ignored.

## 9656 STDIN

9657           The standard input is a text file used only in the following cases:

- 9658           • A prompt is issued for the *-m* or *-y* options.
- 9659           • The *file* operand is specified as *-*.

## 9660 INPUT FILES

9661           Input files are text files whose data is to be included in the SCCS files. If the first character of  
 9662           any line of an input file is SOH (binary 001), the results are unspecified.

## 9663 ENVIRONMENT VARIABLES

9664           The following environment variables affect the execution of *delta*:

9665           *LANG*       Provide a default value for the internationalisation variables that are unset or null. If  
 9666                      *LANG* is unset or null, the corresponding value from the implementation-dependent  
 9667                      default locale will be used. If any of the internationalisation variables contains an  
 9668                      invalid setting, the utility will behave as if none of the variables had been defined.

### 9669 LC\_ALL

9670                      If set to a non-empty string value, override the values of all the other  
 9671                      internationalisation variables.

### 9672 LC\_CTYPE

9673                      Determine the locale for the interpretation of sequences of bytes of text data as  
 9674                      characters (for example, single- as opposed to multi-byte characters in arguments and  
 9675                      input files).

### 9676 LC\_MESSAGES

9677                      Determine the locale that should be used to affect the format and contents of diagnostic  
 9678                      messages written to standard error, and informative messages written to standard  
 9679                      output.

### 9680 NLSPATH

9681                      Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 9682 ASYNCHRONOUS EVENTS

9683           Default.

9684 **STDOUT**

9685 The standard output is used only for the following messages in the POSIX locale:

- 9686 • Prompts (see the
- m**
- and
- y**
- options) in the following formats:

9687 "MRs? "

9688 "comments? "

9689 The MR prompt, if written, always precedes the comments prompt.

- 9690 • A report of each
- file*
- 's activities (unless the
- s**
- option is specified) in the following format:

9691 "%s\n%d inserted\n%d deleted\n%d unchanged\n", <<New SID>, <number  
9692 of lines inserted>, <number of lines deleted>>, <<number of lines  
9693 unchanged>>.9694 **STDERR**

9695 Used only for diagnostic messages.

9696 **OUTPUT FILES**

9697 Any SCCS files updated are files of an unspecified format.

9698 **EXTENDED DESCRIPTION**

9699 None.

9700 **EXIT STATUS**

9701 The following exit values are returned:

9702 0 Successful completion.

9703 &gt;0 An error occurred.

9704 **CONSEQUENCES OF ERRORS**

9705 Default.

9706 **APPLICATION USAGE**

9707 None.

9708 **EXAMPLES**

9709 None.

9710 **FUTURE DIRECTIONS**9711 A version of *delta* that fully supports the **XBD** specification, **Section 10.2, Utility Syntax**  
9712 **Guidelines** may be introduced in a future issue.9713 **SEE ALSO**9714 *admin, diff, get, prs, rmdel.*9715 **CHANGE HISTORY**

9716 First released in Issue 2.

9717 **Issue 4**

9718 Format reorganised.

9719 Exceptions to Utility Syntax Guidelines conformance noted.

9720 Internationalised environment variable support mandated.

9721 **Issue 5**

9722 The output format description in the STDOUT section is corrected.

9723 **NAME**

9724           df — report free disk space

9725 **SYNOPSIS**9726 EX           df [-k][ -P|-t][*file...*]9727 **DESCRIPTION**

9728 EX           The *df* utility writes the amount of available space and file slots for file systems on which the  
 9729           invoking user has appropriate read access. File systems are specified by the *file* operands; when  
 9730           none are specified, information is written for all file systems. The format of the default output  
 9731           from *df* is unspecified, but all space figures will be reported in 512-byte units, unless the **-k**  
 9732           option is specified. This output contains at least the file system names, amount of available  
 9733 EX           space on each of these file systems, and the number of free file slots, or *inodes*, available; when **-t**  
 9734           is specified, the output contains the total allocated space as well.

9735 **OPTIONS**9736           The *df* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

9737           The following options are supported:

9738           **-k**       Use 1024-byte units, instead of the default 512-byte units, when writing space figures.9739           **-P**       Produce output in the format described in the STDOUT section.9740 EX           **-t**       Include total allocated-space figures in the output.9741 **OPERANDS**

9742           The following operand is supported:

9743           *file*       A pathname of a file within the hierarchy of the desired file system. If a file other than  
 9744 EX           a FIFO, a regular file, a directory or a special file representing the device containing the  
 9745           file system (for example, **/dev/dsk/0s1**) is specified, the results are unspecified.  
 9746           Otherwise, *df* will write the amount of free space in the file system containing the  
 9747           specified *file* operand.

9748 **STDIN**

9749           Not used.

9750 **INPUT FILES**

9751           None.

9752 **ENVIRONMENT VARIABLES**9753           The following environment variables affect the execution of *df*:

9754           **LANG**    Provide a default value for the internationalisation variables that are unset or null. If  
 9755           **LANG** is unset or null, the corresponding value from the implementation-dependent  
 9756           default locale will be used. If any of the internationalisation variables contains an  
 9757           invalid setting, the utility will behave as if none of the variables had been defined.

9758           **LC\_ALL**

9759           If set to a non-empty string value, override the values of all the other  
 9760           internationalisation variables.

9761           **LC\_CTYPE**

9762           Determine the locale for the interpretation of sequences of bytes of text data as  
 9763           characters (for example, single- as opposed to multi-byte characters in arguments).

9764 **LC\_MESSAGES**  
 9765 Determine the locale that should be used to affect the format and contents of diagnostic  
 9766 messages written to standard error and informative messages written to standard  
 9767 output.

9768 EX **NLSPATH**  
 9769 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

9770 **ASYNCHRONOUS EVENTS**  
 9771 Default.

9772 **STDOUT**  
 9773 When both the **-k** and **-P** options are specified, the following header line will be written (in the  
 9774 POSIX locale):

9775 "Filesystem 1024-blocks Used Available Capacity Mounted on\n"

9776 When the **-P** option is specified without the **-k** option, the following header line will be written  
 9777 (in the POSIX locale):

9778 "Filesystem 512-blocks Used Available Capacity Mounted on\n"

9779 The implementation may adjust the spacing of the header line and the individual data lines so  
 9780 that the information is presented in orderly columns.

9781 The remaining output with **-P** will consist of one line of information for each specified file  
 9782 system. These lines are formatted as follows:

9783 "%s %d %d %d %d%% %s\n", <file system name>, <total space>,  
 9784 <space used>, <space free>, <percentage used>, <file system root>

9785 In the following list, all quantities expressed in 512-byte units (1024-byte when **-k** is specified)  
 9786 will be rounded up to the next higher unit. The fields are:

9787 <file system name>  
 9788 The name of the file system, in an implementation-dependent format.

9789 <total space>  
 9790 The total size of the file system in 512-byte units. The exact meaning of this figure is  
 9791 implementation-dependent, but should include <space used>, <space free>, plus any  
 9792 space reserved by the system not normally available to a user.

9793 <space used>  
 9794 The total amount of space allocated to existing files in the file system, in 512-byte units.

9795 <space free>  
 9796 The total amount of space available within the file system for the creation of new files  
 9797 by unprivileged users, in 512-byte units. When this figure is less than or equal to zero,  
 9798 it is not possible to create any new files on the file system without first deleting others,  
 9799 unless the process has appropriate privileges. The figure written may be less than zero.

9800 <percentage used>  
 9801 The percentage of the normally available space that is currently allocated to all files on  
 9802 the file system. This is calculated using the fraction:

9803 
$$\text{<space used>}/(\text{<space used>}+\text{<space free>})$$

9804 expressed as a percentage. This percentage may be greater than 100 if <space free> is  
 9805 less than zero. The percentage value is expressed as a positive integer, with any  
 9806 fractional result causing it to be rounded to the next highest integer.

9807           <file system root>  
 9808           The directory below which the file system hierarchy appears.

9809 EX       The output format is unspecified when **-t** is used.

9810 **STDERR**  
 9811       Used only for diagnostic messages.

9812 **OUTPUT FILES**  
 9813       None.

9814 **EXTENDED DESCRIPTION**  
 9815       None.

9816 **EXIT STATUS**  
 9817       The following exit values are returned:  
 9818       0   Successful completion.  
 9819       >0   An error occurred.

9820 **CONSEQUENCES OF ERRORS**  
 9821       Default.

9822 **APPLICATION USAGE**  
 9823       On most systems, the “name of the file system, in an implementation-dependent format” will be  
 9824       the special file on which the file system is mounted. |  
 9825       On large file systems, the calculation specified for percentage used can create huge rounding  
 9826       errors. |

9827 **EXAMPLES**  
 9828       1. The following example writes portable information about the **/usr** file system:  
 9829           df -P /usr  
 9830       2. Assuming that **/usr/src** is part of the **/usr** file system, the following will do the same as the  
 9831       previous example:  
 9832           df -P /usr/src

9833 **FUTURE DIRECTIONS**  
 9834       None.

9835 **SEE ALSO**  
 9836       *find*.

9837 **CHANGE HISTORY**  
 9838       First released in Issue 2. |

9839 **Issue 4**  
 9840       Aligned with the ISO/IEC 9945-2: 1993 standard.

## 9841 NAME

9842 diff — compare two files

## 9843 SYNOPSIS

9844 EX diff [-c | -e | -f | -C n][-br] file1 file2

## 9845 DESCRIPTION

9846 The *diff* utility will compare the contents of *file1* and *file2* and write to standard output a list of  
 9847 changes necessary to convert *file1* into *file2*. This list should be minimal. No output will be  
 9848 produced if the files are identical.

## 9849 OPTIONS

9850 The *diff* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

9851 The following options are supported:

9852 **-b** Cause any amount of white space at the end of a line to be treated as a single newline  
 9853 character (that is, the white-space characters preceding the newline character are  
 9854 ignored) and other strings of white-space characters, not including newline characters,  
 9855 to compare equal.

9856 **-c** Produce output in a form that provides three lines of context.

9857 **-C n** Produce output in a form that provides *n* lines of context (where *n* will be interpreted  
 9858 as a positive decimal integer).

9859 **-e** Produce output in a form suitable as input for the *ed* utility, which can then be used to  
 9860 convert *file1* into *file2*.

9861 EX **-f** Produce output in an alternative form, similar in format to **-e**, but unsuitable as input  
 9862 for the *ed* utility, and in the opposite order.

9863 **-r** Apply *diff* recursively to files and directories of the same name when *file1* and *file2* are  
 9864 both directories.

## 9865 OPERANDS

9866 The following operands are supported:

9867 *file1*

9868 *file2* A pathname of a file to be compared. If either the *file1* or *file2* operand is "-", the standard  
 9869 input will be used in its place.

9870 If both *file1* and *file2* are directories, *diff* will not compare block special files, character special  
 9871 files or FIFO special files to any files and will not compare regular files to directories. The  
 9872 system documentation will specify the behaviour of *diff* on implementation-dependent file types  
 9873 not specified by the **XSH** specification when found in directories. Further details are as specified  
 9874 in **Diff Directory Comparison Format** on page 273.

9875 If only one of *file1* and *file2* is a directory, *diff* will be applied to the non-directory file and the file  
 9876 contained in the directory file with a filename that is the same as the last component of the non-  
 9877 directory file.

## 9878 STDIN

9879 The standard input will be used only if one of the *file1* or *file2* operands references standard  
 9880 input. See the **INPUT FILES** section.

## 9881 INPUT FILES

9882 The input files must be text files.



9883 **ENVIRONMENT VARIABLES**9884 The following environment variables affect the execution of *diff*.

9885 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 9886 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 9887 default locale will be used. If any of the internationalisation variables contains an  
 9888 invalid setting, the utility will behave as if none of the variables had been defined.

9889 **LC\_ALL**

9890 If set to a non-empty string value, override the values of all the other  
 9891 internationalisation variables.

9892 **LC\_CTYPE**

9893 Determine the locale for the interpretation of sequences of bytes of text data as  
 9894 characters (for example, single- as opposed to multi-byte characters in arguments and  
 9895 input files).

9896 **LC\_MESSAGES**

9897 Determine the locale that should be used to affect the format and contents of diagnostic  
 9898 messages written to standard error and informative messages written to standard  
 9899 output.

9900 **LC\_TIME**

9901 Determine the locale for affecting the format of file timestamps written with the **-C** and  
 9902 **-c** options.

9903 EX **NLSPATH**9904 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

9905 **TZ** Determine the locale for affecting the timezone used for calculating file timestamps  
 9906 written with the **-C** and **-c** options.

9907 **ASYNCHRONOUS EVENTS**

9908 Default.

9909 **STDOUT**9910 **Diff Directory Comparison Format**9911 If both *file1* and *file2* are directories, the following output formats will be used.

9912 In the POSIX locale, each file that is present in only one directory will be reported using the  
 9913 following format:

9914 "Only in %s: %s\n", &lt;directory pathname&gt;, &lt;filename&gt;

9915 In the POSIX locale, subdirectories that are common to the two directories may be reported with  
 9916 the following format:

9917 "Common subdirectories: %s and %s\n", <directory1 pathname>,  
 9918 <directory2 pathname>

9919 For each file common to the two directories if the two files are not to be compared, the following  
 9920 format shall be used in the POSIX locale:

9921 "File %s is a %s while file %s is a %s\n",  
 9922 <directory1 pathname>, <file type of directory1 pathname>,  
 9923 <directory2 pathname>, <file type of directory2 pathname>

9924 For each file common to the two directories, if the files are to be compared and are identical, no  
 9925 output shall be written. If the two files differ, the following format shall be written:

9926 "diff %s %s %s\n", <diff\_options>, <filename1>, <filename2>

9927 where <diff\_options> are the options as specified on the command line. Depending on these  
 9928 options, one of the following output formats will be used to write the differences.

9929 All directory pathnames listed in this section will be relative to the original command line  
 9930 arguments. All other names of files listed in this section will be filenames (pathname  
 9931 components).

### 9932 Diff Default Output Format

9933 EX The default (without **-e**, **-f**, **-c** or **-C** options) *diff* utility output contains lines of these forms:

9934 "%da%d\n", <num1>, <num2>

9935 "%da%d,%d\n", <num1>, <num2>, <num3>

9936 "%dd%d\n", <num1>, <num2>

9937 "%d,%dd%d\n", <num1>, <num2>, <num3>

9938 "%dc%d\n", <num1>, <num2>

9939 "%d,%dc%d\n", <num1>, <num2>, <num3>

9940 "%dc%d,%d\n", <num1>, <num2>, <num3>

9941 "%d,%dc%d,%d\n", <num1>, <num2>, <num3>, <num4>

9942 These lines resemble *ed* subcommands to convert *file1* into *file2*. The line numbers before the  
 9943 action letters pertain to *file1*; those after pertain to *file2*. Thus, by exchanging a for d and reading  
 9944 the line in reverse order, one can also determine how to convert *file2* into *file1*. As in *ed*, identical  
 9945 pairs (where *num1* = *num2*) are abbreviated as a single number.

9946 Following each of these lines, *diff* will write to standard output all lines affected in the first file  
 9947 using the format:

9948 "<A%S", <line>

9949 and all lines affected in the second file using the format:

9950 ">A%S", <line>

9951 If there are lines affected in both *file1* and *file2* (as with the **c** subcommand), the changes are  
 9952 separated with a line consisting of three hyphens:

9953 "---\n"

### 9954 Diff -e Output Format

9955 With the **-e** option, a script will be produced that will, when provided as input to *ed*, along with  
 9956 an appended **w** (write) command, convert *file1* into *file2*. Only the **a** (append), **c** (change), **d**  
 9957 (delete), **i** (insert), and **s** (substitute) commands of *ed* will be used in this script. Text lines, except  
 9958 those consisting of the single character period (.), will be output as they appear in the file.

**Diff -f Output Format**

With the **-f** option, an alternative format of script will be produced. It will be similar to that produced by **-e**, with the following differences:

1. It will be expressed in reverse sequence; the output of **-e** will order changes from the end of the file to the beginning; the **-f** from beginning to end.
2. The command form **<lines> <command-letter>** used by **-e** will be reversed. For example, **10c** with **-e** would be **c10** with **-f**.
3. The form used for ranges of line numbers will be space-character-separated, rather than comma-separated.

**Diff -c or -C Output Format**

With the **-c** or **-C** option, the output format will consist of affected lines along with surrounding lines of context. The affected lines will show which ones need to be deleted or changed in *file1*, and those added from *file2*. With the **-c** option, three lines of context, if available, will be written before and after the affected lines. With the **-C** option, the user can specify how many lines of context will be written. The exact format follows.

The name and last modification time of each file will be output in the following format:

```
**** %s %s\n", file1, <file1 time stamp> "--- %s %s\n", file2,
<file2 time stamp>
```

and a string of 15 asterisks:

```
"*****\n"
```

Each **<file>** field will be the pathname of the corresponding file being compared. The pathname written for standard input is unspecified.

In the POSIX locale, each **<timestamp>** field will be equivalent to the output from the following command:

```
date "+%a %b %e %T %Y"
```

without the trailing newline character, executed at the time of last modification of the corresponding file (or the current time, if the file is standard input).

Then, the following output formats will be applied for every set of changes.

First, the range of lines in *file1* will be written in the following format:

```
**** %d,%d ****\n", <beginning line number>, <ending line number>
```

Next, the affected lines along with lines of context (unaffected lines) will be written. Unaffected lines will be written in the following format:

```
"ΔΔ%S", <unaffected_line>
```

Deleted lines will be written as:

```
"-Δ%S", <deleted_line>
```

Changed lines will be written as:

```
"!Δ%S", <changed_line>
```

9996 Next, the range of lines in *file2* will be written in the following format:

9997 " --- %d,%d ----\n", <beginning line number>, <ending line number>

9998 Then, lines of context and changed lines will be written as described in the previous formats.

9999 Lines added from *file2* will be written in the following format:

10000 "+Δ%\$ ", <added\_line>

10001 **STDERR**

10002 Used only for diagnostic messages.

10003 **OUTPUT FILES**

10004 None.

10005 **EXTENDED DESCRIPTION**

10006 None.

10007 **EXIT STATUS**

10008 The following exit values are returned:

10009 0 No differences were found.

10010 1 Differences were found.

10011 >1 An error occurred.

10012 **CONSEQUENCES OF ERRORS**

10013 Default.

10014 **APPLICATION USAGE**

10015 If lines at the end of a file are changed and other lines are added, *diff* output may show this as a

10016 delete and add, as a change, or as a change and add; *diff* is not expected to know which

10017 happened and users should not care about the difference in output as long as it clearly shows the

10018 differences between the files.

10019 **EXAMPLES**

10020 If **dir1** is a directory containing a directory named **x**, **dir2** is a directory containing a directory

10021 named **x**, **dir1/x** and **dir2/x** both contain files named **date.out**, and **dir2/x** contains a file named **y**,

10022 the command:

10023 `diff -r dir1 dir2`

10024 could produce output similar to:

10025 Common subdirectories: dir1/x and dir2/x

10026 Only in dir2/x: y

10027 `diff -r dir1/x/date.out dir2/x/date.out`

10028 `lcl`

10029 `< Mon Jul 2 13:12:16 PDT 1990`

10030 `---`

10031 `> Tue Jun 19 21:41:39 PDT 1990`

10032 **FUTURE DIRECTIONS**

10033 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this

10034 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the

10035 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when

10036 finalised.

10037 **SEE ALSO**

10038 *cmp, comm, dircmp, ed.*

10039 **CHANGE HISTORY**

10040 First released in Issue 2.

10041 **Issue 4**

10042 Aligned with the ISO/IEC 9945-2: 1993 standard.

10043 **Issue 5**

10044 FUTURE DIRECTIONS section added.

10045 **NAME**10046           dircmp — directory comparison (**LEGACY**)10047 **SYNOPSIS**10048 EX       dircmp [-ds] *dir1 dir2*10049 **DESCRIPTION**

10050       The *dircmp* utility examines the directory hierarchies specified by *dir1* and *dir2* and generates  
 10051       various tabulated information about the contents of the directories. Sorted listings of files that  
 10052       are unique to each directory are generated for all the options. If no option is specified, a list is  
 10053       output that indicates whether the filenames common to both directories have the same contents.

10054 **OPTIONS**

10055       The *dircmp* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 10056       following options are supported:

10057       **-d**       Compare the contents of files with the same name in both directories and output a list  
 10058       indicating what must be changed in the two files to bring them into agreement. The list  
 10059       format is described in *diff*.

10060       **-s**       Suppress messages about identical files.

10061 **OPERANDS**

10062       The following operands are supported:

10063       *dir1*  
 10064       *dir2*     A pathname of a directory to be compared.

10065 **STDIN**

10066       Not used.

10067 **INPUT FILES**

10068       None.

10069 **ENVIRONMENT VARIABLES**

10070       The following environment variables may affect the execution of *dircmp*:

10071       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 10072       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 10073       default locale will be used. If any of the internationalisation variables contains an  
 10074       invalid setting, the utility will behave as if none of the variables had been defined.

10075       **LC\_ALL**

10076       If set to a non-empty string value, override the values of all the other  
 10077       internationalisation variables.

10078       **LC\_COLLATE**

10079       Determine the locale for the ordering of the output.

10080       **LC\_CTYPE**

10081       Determine the locale for the interpretation of sequences of bytes of text data as  
 10082       characters (for example, single- as opposed to multi-byte characters in arguments and  
 10083       input files).

10084       **LC\_MESSAGES**

10085       Determine the locale that should be used to affect the format and contents of diagnostic  
 10086       messages written to standard error.

10087       **NLSPATH**

10088       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

10089 **ASYNCHRONOUS EVENTS**

10090 Default.

10091 **STDOUT**10092 The output format of *dircmp* is unspecified, but includes:

- 10093 • lists of pathnames in either *dir1* or *dir2*, but not both
- 10094 • when **-d** is given, listings of differences between files, as produced by *diff*.

10095 **STDERR**

10096 Used only for diagnostic messages.

10097 **OUTPUT FILES**

10098 None.

10099 **EXTENDED DESCRIPTION**

10100 None.

10101 **EXIT STATUS**

10102 The following exit values are returned:

- 10103 0 Successful completion.
- 10104 >0 An error occurred. (Differences in directory contents are not considered errors.)

10105 **CONSEQUENCES OF ERRORS**

10106 Default.

10107 **APPLICATION USAGE**10108 Applications should migrate to the *diff -r* command.10109 **EXAMPLES**

10110 None.

10111 **FUTURE DIRECTIONS**

10112 None.

10113 **SEE ALSO**10114 *cmp*, *diff*.10115 **CHANGE HISTORY**

10116 First released in Issue 2.

10117 **Issue 4**

10118 Format reorganised.

10119 Internationalised environment variable support made optional.

10120 Marked TO BE WITHDRAWN.

10121 **Issue 5**

10122 Marked LEGACY.

10123 **NAME**

10124           dirname — return the directory portion of pathname

10125 **SYNOPSIS**10126           dirname *string*10127 **DESCRIPTION**

10128       The *string* operand will be treated as a pathname, as defined in **pathname** (see the **XBD**  
 10129       specification, **Chapter 2, Glossary**). The string *string* will be converted to the name of the  
 10130       directory containing the filename corresponding to the last pathname component in *string*,  
 10131       performing actions equivalent to the following steps in order:

- 10132           1. If *string* is //, skip steps 2 to 5.
- 10133           2. If *string* consists entirely of slash characters, *string* will be set to a single slash character. In  
 10134           this case, skip steps 3 to 8.
- 10135           3. If there are any trailing slash characters in *string*, they will be removed.
- 10136           4. If there are no slash characters remaining in *string*, *string* will be set to a single period  
 10137           character. In this case, skip steps 5 to 8.
- 10138           5. If there are any trailing non-slash characters in *string*, they will be removed.
- 10139           6. If the remaining *string* is //, it is implementation-dependent whether steps 7 and 8 are  
 10140           skipped or processed.
- 10141           7. If there are any trailing slash characters in *string*, they will be removed.
- 10142           8. If the remaining *string* is empty, *string* will be set to a single slash character.

10143       The resulting string will be written to standard output.

10144 **OPTIONS**

10145       None.

10146 **OPERANDS**

10147       The following operand is supported:

10148       *string*   A string.10149 **STDIN**

10150       Not used.

10151 **INPUT FILES**

10152       None.

10153 **ENVIRONMENT VARIABLES**10154       The following environment variables affect the execution of *dirname*:

10155       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 10156       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 10157       default locale will be used. If any of the internationalisation variables contains an  
 10158       invalid setting, the utility will behave as if none of the variables had been defined.

10159       **LC\_ALL**

10160       If set to a non-empty string value, override the values of all the other  
 10161       internationalisation variables.

10162       **LC\_CTYPE**

10163       Determine the locale for the interpretation of sequences of bytes of text data as  
 10164       characters (for example, single- as opposed to multi-byte characters in arguments).



10165 **LC\_MESSAGES**  
 10166 Determine the locale that should be used to affect the format and contents of diagnostic  
 10167 messages written to standard error.

10168 EX **NLSPATH**  
 10169 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 10170 ASYNCHRONOUS EVENTS

10171 Default.

## 10172 STDOUT

10173 The *dirname* utility will write a line to the standard output in the following format:

10174 "%s\n", *<resulting string>*

## 10175 STDERR

10176 Used only for diagnostic messages.

## 10177 OUTPUT FILES

10178 None.

## 10179 EXTENDED DESCRIPTION

10180 None.

## 10181 EXIT STATUS

10182 The following exit values are returned:

10183 0 Successful completion.

10184 >0 An error occurred.

## 10185 CONSEQUENCES OF ERRORS

10186 Default.

## 10187 APPLICATION USAGE

10188 The definition of *pathname* specifies implementation-dependent behaviour for pathnames  
 10189 starting with two slash characters. Therefore, applications must not arbitrarily add slashes to  
 10190 the beginning of a pathname unless they can ensure that there are more or less than two or are  
 10191 prepared to deal with the implementation-dependent consequences.

## 10192 EXAMPLES

10193

10194

10195

10196

10197

10198

10199

10200

10201

10202

10203

10204

| Command                       | Results                  |
|-------------------------------|--------------------------|
| <code>dirname /</code>        | <code>/</code>           |
| <code>dirname //</code>       | <code>/ or //</code>     |
| <code>dirname /a/b/</code>    | <code>/a</code>          |
| <code>dirname //a//b//</code> | <code>//a</code>         |
| <code>dirname</code>          | <i>unspecified</i>       |
| <code>dirname a</code>        | <code>. (\$? = 0)</code> |
| <code>dirname ""</code>       | <code>. (\$? = 0)</code> |
| <code>dirname /a</code>       | <code>/</code>           |
| <code>dirname /a/b</code>     | <code>/a</code>          |
| <code>dirname a/b</code>      | <code>a</code>           |

## 10205 FUTURE DIRECTIONS

10206 None.

## 10207 SEE ALSO

10208 *basename*, Section 2.5 on page 27.

10209 **CHANGE HISTORY**

10210 First released in Issue 2.

10211 **Issue 4**

10212 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 10213 NAME

10214       dis — disassembler (**DEVELOPMENT, LEGACY**)

## 10215 SYNOPSIS

10216 **PI**       dis [-oLV][-F *function*...] [-l *string*] *file*...

## 10217 DESCRIPTION

10218       The *dis* utility produces an assembly language listing of each of its *file* arguments, each of which  
 10219       may be an object file or an archive of object files. The listing includes assembly statements and  
 10220       an octal or hexadecimal representation of the binary that produced those statements.

## 10221 OPTIONS

10222       The *dis* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 10223       following options are supported:

10224       **-o**       Write numbers in octal. The default is hexadecimal.

10225       **-L**       Invoke a lookup of C-language source labels in the symbol table for subsequent writing  
 10226       to standard output.

10227       **-V**       Write the version number of the disassembler to standard error.

10228       **-F *function***

10229               Disassemble only the named *function* in each object file specified on the command line.

10230       **-l *string***

10231               Disassemble the library file specified as *string*. For example, the command *dis -l m*  
 10232               will disassemble the math library.

## 10233 OPERANDS

10234       The following operands are supported:

10235       *file*       A pathname of an object file or an archive (see *ar*) of object files.

## 10236 STDIN

10237       Not used.

## 10238 INPUT FILES

10239       The input files are object files or archives of object files, or both.

## 10240 ENVIRONMENT VARIABLES

10241       The following environment variables may affect the execution of *dis*:

10242       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 10243       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 10244       default locale will be used. If any of the internationalisation variables contains an  
 10245       invalid setting, the utility will behave as if none of the variables had been defined.

10246       **LC\_ALL**

10247               If set to a non-empty string value, override the values of all the other  
 10248               internationalisation variables.

10249       **LC\_CTYPE**

10250               Determine the locale for the interpretation of sequences of bytes of text data as  
 10251               characters (for example, single- as opposed to multi-byte characters in arguments and  
 10252               input files).

10253       **LC\_MESSAGES**

10254               Determine the locale that should be used to affect the format and contents of diagnostic  
 10255               messages written to standard error.

10256 *NLSPATH*  
10257 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

10258 **ASYNCHRONOUS EVENTS**  
10259 Default.

10260 **STDOUT**  
10261 The standard output consists of an assembly listing of unspecified format.

10262 **STDERR**  
10263 Used only for diagnostic messages.

10264 **OUTPUT FILES**  
10265 None.

10266 **EXTENDED DESCRIPTION**  
10267 None.

10268 **EXIT STATUS**  
10269 The following exit values are returned:  
10270 0 Successful completion.  
10271 >0 An error occurred.

10272 **CONSEQUENCES OF ERRORS**  
10273 Default.

10274 **APPLICATION USAGE**  
10275 None.

10276 **EXAMPLES**  
10277 None.

10278 **FUTURE DIRECTIONS**  
10279 None.

10280 **SEE ALSO**  
10281 *ar, cc, c89*.

10282 **CHANGE HISTORY**  
10283 First released in Issue 2.

10284 **Issue 4**  
10285 Format reorganised.  
10286 Utility Syntax Guidelines support mandated.  
10287 Internationalised environment variable support made optional.  
10288 Marked TO BE WITHDRAWN.

10289 **Issue 5**  
10290 Marked LEGACY.

10291 **NAME**10292        **du** — estimate file space usage (**LEGACY**)10293 **SYNOPSIS**10294 EX OB    **du** [-a] [-s] [-kx] [-r] [*file* ...]10295 **DESCRIPTION**

10296        By default, the *du* utility writes to standard output the size of the file space allocated to, and the  
 10297        size of the file space allocated to each subdirectory of, the file hierarchy rooted in each of the  
 10298        specified files. The size of the file space allocated to a file of type directory is defined as the sum  
 10299        total of space allocated to all files in the file hierarchy rooted in the directory plus the space  
 10300        allocated to the directory itself.

10301        When *du* cannot *stat()* files or *stat()* or read directories, it will report an error condition and the  
 10302        final exit status will be affected. Files with multiple links will be counted and written for only  
 10303        one entry. The directory entry that is selected in the report is unspecified. By default, file sizes  
 10304        are written in 512-byte units, rounded up to the next 512-byte unit.

10305 **OPTIONS**10306        The *du* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

10307        The following options are supported:

10308        **-a**        In addition to the default output, report the size of each file not of type directory in the  
 10309        file hierarchy rooted in the specified file. Regardless of the presence of the **-a** option,  
 10310        non-directories given as *file* operands will always be listed.

10311        **-k**        Write the files sizes in units of 1024 bytes, rather than the default 512-byte units.

10312 EX OB    **-r**        Generate messages about directories that cannot be read, files that cannot be opened,  
 10313        and so on. This is the default case.

10314        **-s**        Instead of the default output, report only the total sum for each of the specified files.

10315        **-x**        When evaluating file sizes, evaluate only those files that have the same device as the  
 10316        file specified by the *file* operand.

10317 **OPERANDS**

10318        The following operand is supported:

10319        *file*        The pathname of a file whose size is to be written. If no *file* is specified, the current  
 10320        directory is used.

10321 **STDIN**

10322        Not used.

10323 **INPUT FILES**

10324        None.

10325 **ENVIRONMENT VARIABLES**10326        The following environment variables affect the execution of *du*:

10327        **LANG**        Provide a default value for the internationalisation variables that are unset or null. If  
 10328        **LANG** is unset or null, the corresponding value from the implementation-dependent  
 10329        default locale will be used. If any of the internationalisation variables contains an  
 10330        invalid setting, the utility will behave as if none of the variables had been defined.

10331        **LC\_ALL**

10332        If set to a non-empty string value, override the values of all the other  
 10333        internationalisation variables.

|          |                                                 |                                                                                                       |
|----------|-------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| 10334    | <i>LC_CTYPE</i>                                 |                                                                                                       |
| 10335    |                                                 | Determine the locale for the interpretation of sequences of bytes of text data as                     |
| 10336    |                                                 | characters (for example, single- as opposed to multi-byte characters in arguments).                   |
| 10337    | <i>LC_MESSAGES</i>                              |                                                                                                       |
| 10338    |                                                 | Determine the locale that should be used to affect the format and contents of diagnostic              |
| 10339    |                                                 | messages written to standard error.                                                                   |
| 10340 EX | <i>NLSPATH</i>                                  |                                                                                                       |
| 10341    |                                                 | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .               |
| 10342    | <b>ASYNCHRONOUS EVENTS</b>                      |                                                                                                       |
| 10343    | Default.                                        |                                                                                                       |
| 10344    | <b>STDOUT</b>                                   |                                                                                                       |
| 10345    |                                                 | The output from <i>du</i> consists of the amount of the space allocated to a file and the name of the |
| 10346    |                                                 | file, in the following format:                                                                        |
| 10347    | "%d %s\n", <size>, <pathname>                   |                                                                                                       |
| 10348    | <b>STDERR</b>                                   |                                                                                                       |
| 10349    | Used only for diagnostic messages.              |                                                                                                       |
| 10350    | <b>OUTPUT FILES</b>                             |                                                                                                       |
| 10351    | None.                                           |                                                                                                       |
| 10352    | <b>EXTENDED DESCRIPTION</b>                     |                                                                                                       |
| 10353    | None.                                           |                                                                                                       |
| 10354    | <b>EXIT STATUS</b>                              |                                                                                                       |
| 10355    | The following exit values are returned:         |                                                                                                       |
| 10356    | 0                                               | Successful completion.                                                                                |
| 10357    | >0                                              | An error occurred.                                                                                    |
| 10358    | <b>CONSEQUENCES OF ERRORS</b>                   |                                                                                                       |
| 10359    | Default.                                        |                                                                                                       |
| 10360    | <b>APPLICATION USAGE</b>                        |                                                                                                       |
| 10361    | None.                                           |                                                                                                       |
| 10362    | <b>EXAMPLES</b>                                 |                                                                                                       |
| 10363    | None.                                           |                                                                                                       |
| 10364    | <b>FUTURE DIRECTIONS</b>                        |                                                                                                       |
| 10365    | None.                                           |                                                                                                       |
| 10366    | <b>SEE ALSO</b>                                 |                                                                                                       |
| 10367    | <i>ls</i> .                                     |                                                                                                       |
| 10368    | <b>CHANGE HISTORY</b>                           |                                                                                                       |
| 10369    | First released in Issue 2.                      |                                                                                                       |
| 10370    | <b>Issue 4</b>                                  |                                                                                                       |
| 10371    | Aligned with the ISO/IEC 9945-2: 1993 standard. |                                                                                                       |
| 10372    | <b>Issue 5</b>                                  |                                                                                                       |
| 10373    | Marked LEGACY.                                  |                                                                                                       |

10374 **NAME**

10375 echo — write arguments to standard output

10376 **SYNOPSIS**10377 echo [*string* ...]10378 **DESCRIPTION**

10379 The *echo* utility will write its arguments to standard output, followed by a newline character. If  
 10380 there are no arguments, only the newline character will be written.

10381 **OPTIONS**

10382 The *echo* utility will not recognise the `--` argument in the manner specified by Guideline 10 of  
 10383 the XBD specification, **Section 10.2, Utility Syntax Guidelines**; `--` will be recognised as a string  
 10384 operand.

10385 Implementations will not support any options.

10386 **OPERANDS**

10387 The following operands are supported:

10388 EX *string* A string to be written to standard output. If any operand is `"-n"`, it will be treated as a  
 10389 string, not an option. The following character sequences will be recognised within any  
 10390 of the arguments:

10391 `\a` Write an alert character.

10392 `\b` Write a backspace character.

10393 `\c` Suppress the newline character that otherwise follows the final argument in  
 10394 the output. All characters following the `\c` in the arguments will be ignored.

10395 `\f` Write a form-feed character.

10396 `\n` Write a newline character.

10397 `\r` Write a carriage-return character.

10398 `\t` Write a tab character.

10399 `\v` Write a vertical-tab character.

10400 `\\` Write a backslash character.

10401 `\0num` Write an 8-bit value that is the zero-, one-, two- or three-digit octal number  
 10402 *num*.

10403 **STDIN**

10404 Not used.

10405 **INPUT FILES**

10406 None.

10407 **ENVIRONMENT VARIABLES**10408 The following environment variables affect the execution of *echo*:

10409 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 10410 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 10411 default locale will be used. If any of the internationalisation variables contains an  
 10412 invalid setting, the utility will behave as if none of the variables had been defined.

10413 **LC\_ALL**  
 10414 If set to a non-empty string value, override the values of all the other  
 10415 internationalisation variables.

10416 EX **LC\_CTYPE**  
 10417 Determine the locale for the interpretation of sequences of bytes of text data as  
 10418 characters (for example, single- as opposed to multi-byte characters in arguments).

10419 **LC\_MESSAGES**  
 10420 Determine the locale that should be used to affect the format and contents of diagnostic  
 10421 messages written to standard error.

10422 EX **NLSPATH**  
 10423 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

10424 **ASYNCHRONOUS EVENTS**  
 10425 Default.

10426 **STDOUT**  
 10427 The *echo* utility arguments will be separated by single space characters and a newline character  
 10428 EX will follow the last argument. Output transformations will occur based on the escape sequences  
 10429 in the input; see the OPERANDS section.

10430 **STDERR**  
 10431 Used only for diagnostic messages.

10432 **OUTPUT FILES**  
 10433 None.

10434 **EXTENDED DESCRIPTION**  
 10435 None.

10436 **EXIT STATUS**  
 10437 The following exit values are returned:  
 10438 0 Successful completion.  
 10439 >0 An error occurred.

10440 **CONSEQUENCES OF ERRORS**  
 10441 Default.

10442 **APPLICATION USAGE**  
 10443 It is not possible to use *echo* portably across all systems that are not XSI-conformant unless both  
 10444 **-n** (as the first argument) and escape sequences are omitted.

10445 The *printf* utility can be used portably to emulate any of the traditional behaviours of the *echo*  
 10446 utility as follows:

10447 • The XSI *echo* is equivalent to:  
 10448 `printf "%b\n" "$*"`



- 10449           • The BSD *echo* is equivalent to:

```
10450 if ["X$1" = "X-n"]
10451 then
10452 shift
10453 printf "%s" "$*"
10454 else
10455 printf "%s\n" "$*"
10456 fi
```

10457           New applications are encouraged to use *printf* instead of *echo*.

10458 **EXAMPLES**

10459           None.

10460 **FUTURE DIRECTIONS**

10461           None.

10462 **SEE ALSO**

10463           *printf*.

10464 **CHANGE HISTORY**

10465           First released in Issue 2.

10466 **Issue 4**

10467           Aligned with the ISO/IEC 9945-2: 1993 standard.

10468 **Issue 5**

10469           In the OPTIONS section, the last sentence is changed to indicate that implementations "will not"  
10470           support any options; in the previous issue this said "need not".

## 10471 NAME

10472 ed — edit text

## 10473 SYNOPSIS

10474 ed [-p *string*][-s][*file*]10475 OB ed [-p *string*][-][*file*]

## 10476 DESCRIPTION

10477 The *ed* utility is a line-oriented text editor that uses two modes: *command mode* and *input mode*.  
 10478 In command mode the input characters are interpreted as commands, and in input mode they  
 10479 are interpreted as text. See the EXTENDED DESCRIPTION section.

## 10480 OPTIONS

10481 OB The *ed* utility supports the XBD specification, Section 10.2, Utility Syntax Guidelines, except  
 10482 for its non-standard usage of "-".

10483 The following options are supported:

10484 -p *string*

10485 Use *string* as the prompt string when in command mode. By default, there is no  
 10486 prompt string.

10487 -s Suppress the writing of byte counts by **e**, **E**, **r** and **w** commands and of the "!" prompt  
 10488 after a *!command*.

10489 OB - Same as the -s option.

## 10490 OPERANDS

10491 The following operand is supported:

10492 *file* If the *file* argument is given, *ed* will simulate an **e** command on the file named by the  
 10493 pathname, *file*, before accepting commands from the standard input.

## 10494 STDIN

10495 The standard input must be a text file consisting of commands, as described in the EXTENDED  
 10496 DESCRIPTION section.

## 10497 INPUT FILES

10498 The input files must be text files.

## 10499 ENVIRONMENT VARIABLES

10500 The following environment variables affect the execution of *ed*:10501 *HOME* Determine the pathname of the user's home directory.

10502 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 10503 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 10504 default locale will be used. If any of the internationalisation variables contains an  
 10505 invalid setting, the utility will behave as if none of the variables had been defined.

10506 *LC\_ALL*

10507 If set to a non-empty string value, override the values of all the other  
 10508 internationalisation variables.

10509 *LC\_COLLATE*

10510 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 10511 character collating elements within regular expressions.

10512 **LC\_CTYPE**

10513 Determine the locale for the interpretation of sequences of bytes of text data as  
 10514 characters (for example, single- as opposed to multi-byte characters in arguments and  
 10515 input files) and the behaviour of character classes within regular expressions.

10516 **LC\_MESSAGES**

10517 Determine the locale that should be used to affect the format and contents of diagnostic  
 10518 messages written to standard error and informative messages written to standard  
 10519 output.

10520 EX **NLSPATH**

10521 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

10522 **ASYNCHRONOUS EVENTS**

10523 The *ed* utility will take the standard action for all signals (see the ASYNCHRONOUS EVENTS  
 10524 section in Section 1.9 on page 11) with the following exceptions:

10525 **SIGINT** The *ed* utility will interrupt its current activity, write the string `?\n` to standard output,  
 10526 and return to command mode (see the EXTENDED DESCRIPTION section).

10527 **SIGHUP**

10528 If the buffer is not empty and has changed since the last write, the *ed* utility will  
 10529 attempt to write a copy of the buffer in a file. First, the file named **ed.hup** in the current  
 10530 directory will be used; if that fails, the file named **ed.hup** in the directory named by the  
 10531 *HOME* environment variable will be used. In any case, the *ed* utility will exit without  
 10532 returning to command mode.

10533 **STDOUT**

10534 Various editing commands and the prompting feature (see **-p**) write to standard output, as  
 10535 described in the EXTENDED DESCRIPTION section.

10536 **STDERR**

10537 Used only for diagnostic messages.

10538 **OUTPUT FILES**

10539 The output files are text files whose formats are dependent on the editing commands given.

10540 **EXTENDED DESCRIPTION**

10541 The *ed* utility operates on a copy of the file it is editing; changes made to the copy will have no  
 10542 effect on the file until a **w** (write) command is given. The copy of the text is called the *buffer*.

10543 Commands to *ed* have a simple and regular structure: zero, one or two *addresses* followed by a  
 10544 single-character *command*, possibly followed by parameters to that command. These addresses  
 10545 specify one or more lines in the buffer. Every command that requires addresses has default  
 10546 addresses, so that the addresses very often can be omitted. If the **-p** option is specified, the  
 10547 prompt string will be written to standard output before each command is read.

10548 In general, only one command can appear on a line. Certain commands allow text to be input.  
 10549 This text is placed in the appropriate place in the buffer. While *ed* is accepting text, it is said to  
 10550 be in *input mode*. In this mode, no commands are recognised; all input is merely collected. Input  
 10551 mode is terminated by entering a line consisting of two characters: a period (.) followed by a  
 10552 newline character. This line is not considered part of the input text.

## 10553 Regular Expressions in ed

10554 The *ed* utility supports basic regular expressions, as described in the **XBD** specification, **Section**  
 10555 **7.3, Basic Regular Expressions**. Since regular expressions in *ed* are always matched against  
 10556 single lines, never against any larger section of text, there is no way for a regular expression to  
 10557 match a newline character. A null RE is equivalent to the last RE encountered.

10558 Regular expressions are used in addresses to specify lines, and in some commands (for example,  
 10559 the **s** substitute command) to specify portions of a line to be substituted.

## 10560 Addresses in ed

10561 Addressing in *ed* relates to the *current line*. Generally, the current line is the last line affected by a  
 10562 command. The *current line number* is the address (line number) of the current line. The exact  
 10563 effect on the current line number is discussed under the description of each command. The **f**, **h**,  
 10564 **H**, **k**, **P**, **w**, **=** and **!** commands do not modify the current line number.

10565 Addresses are constructed as follows:

- 10566 1. The character "." (period) addresses the current line.
- 10567 2. The character "\$" addresses the last line of the buffer.
- 10568 3. A positive decimal number *n* addresses the *n*th line of the buffer. The first line in the  
 10569 buffer is line number 1.
- 10570 4. **mark name character** *x*, which must be a lower-case letter from the portable character set.  
 10571 Lines can be marked with the **k** command.
- 10572 5. An RE enclosed by slashes (/) addresses the first line found by searching forward from the  
 10573 line following the current line toward the end of the buffer and stopping at the first line  
 10574 containing a string matching the RE. An address consisting of a null RE delimited by  
 10575 slashes (/ /) addresses the next line containing the last RE encountered. If necessary, the  
 10576 search will wrap around to the beginning of the buffer and continue up to and including  
 10577 the current line, so that the entire buffer is searched. Within the RE, the sequence \  
 10578 represents a literal slash instead of the RE delimiter.
- 10579 6. An RE enclosed in question-marks (?) addresses the first line found by searching  
 10580 backward from the line preceding the current line toward the beginning of the buffer and  
 10581 stopping at the first line containing a string matching the RE. If necessary, the search  
 10582 wraps around to the end of the buffer and continues up to and including the current line.  
 10583 Within the RE, the sequence \  
 10584 represents a literal question-mark instead of the RE  
 delimiter.
- 10585 7. An address followed by a plus sign (+) or a minus sign (−) followed by a decimal number  
 10586 specifies that address plus (respectively minus) the indicated number of lines. The plus  
 10587 sign can be omitted.
- 10588 8. If an address begins with "+" or "−", the addition or subtraction is taken with respect to the  
 10589 current line number; for example, −5 is understood to mean .−5.
- 10590 9. If an address ends with "+" or "−", then 1 will be added to or subtracted from the address,  
 10591 respectively. As a consequence of this rule and of rule 8 immediately above, the address  
 10592 "−" refers to the line preceding the current line. Moreover, trailing "+" and "−" characters  
 10593 have a cumulative effect, so −− refers to the current line number less 2.
- 10594 10. A comma (,) stands for the address pair 1,\$, while a semicolon (;) stands for the pair .,\$.

10595 Commands require zero, one or two addresses. Commands that require no addresses regard the  
 10596 presence of an address as an error. Commands that accept one or two addresses assume default

10597 addresses when no addresses are given. If one address is given to a command that allows two  
 10598 addresses, the command will operate as if it were specified as:

10599 `given_address; . command`

10600 If more addresses are given than such a command requires, the results are undefined.

10601 Typically, addresses are separated from each other by a comma. They can also be separated by a  
 10602 semicolon. In the latter case, the current line number (.) is set to the first address, and only then  
 10603 will the second address be calculated. This feature can be used to determine the starting line for  
 10604 forward and backward searches (see rules 5 and 6 above). The second address of any two-  
 10605 address sequence corresponds to a line that does not precede, in the buffer, the line  
 10606 corresponding to the first address.

## 10607 **Commands in ed**

10608 In the following list of *ed* commands, the default addresses are shown in parentheses. The  
 10609 number of addresses shown in the default are the number expected by the command. The  
 10610 parentheses are not part of the address; they show that the given addresses are the default.

10611 It is generally invalid for more than one command to appear on a line. However, any command  
 10612 (except **e**, **E**, **f**, **q**, **Q**, **r**, **w** and **!**) can be suffixed by the letter **l**, **n** or **p**; in which case, except for the  
 10613 **l**, **n** and **p** commands, the command will be executed and then the new current line will be  
 10614 written as described below under the **l**, **n** and **p** commands. When an **l**, **n** or **p** suffix is used with  
 10615 an **l**, **n** or **p** command, the command will write to standard output as described below, but it is  
 10616 unspecified whether the suffix writes the current line again in the requested format or whether  
 10617 the suffix has no effect. For example, the **pl** command (base **p** command with an **l** suffix) will  
 10618 either write just the current line or will write it twice once as specified for **p** and once as  
 10619 specified for **l**. Also, the **g**, **G**, **v** and **V** commands takes a command as a parameter.

10620 Each address component can be preceded by zero or more blank characters. The command  
 10621 letter can be preceded by zero or more blank characters. If a suffix letter (**l**, **n** or **p**) is given, it  
 10622 must immediately follow the command.

10623 The **e**, **E**, **f**, **r** and **w** commands take an optional *file* parameter, separated from the command  
 10624 letter by one or more blank characters.

10625 If changes have been made in the buffer since the last **w** command that wrote the entire buffer,  
 10626 *ed* will warn the user if an attempt is made to destroy the editor buffer via the **e** or **q** commands.  
 10627 The *ed* utility will write the string:

10628 `" ?\n"`

10629 (followed by an explanatory message if *help mode* has been enabled via the **H** command) to  
 10630 standard output and will continue in command mode with the current line number unchanged.  
 10631 If the **e** or **q** command is repeated with no intervening command, it will take effect.

10632 If an end-of-file is detected on standard input when a command is expected, the *ed* utility acts as  
 10633 if a **q** command had been entered.

10634 If the closing delimiter of an RE or of a replacement string (for example, **/**) in a **g**, **G**, **s**, **v** or **V**  
 10635 command would be the last character before a newline character, that delimiter can be omitted,  
 10636 in which case the addressed line is written. For example, the following pairs of commands are  
 10637 equivalent:

10638           s/s1/s2 s/s1/s2/p

10639           g/s1    g/s1/p

10640           ?s1     ?s1?

10641       If an invalid command is entered, *ed* will write the string:

10642           "?\\n"

10643       (followed by an explanatory message if *help mode* has been enabled via the **H** command) to  
10644       standard output and will continue in command mode with the current line number unchanged.

## 10645       **Append Command**

10646       *Synopsis:*       ( . )a  
10647                       <text>  
10648                       .

10649       The **a** command reads the given text and appends it after the addressed line; the current line  
10650       number will become the address of the last inserted line or, if there were none, the addressed  
10651       line. Address 0 is valid for this command; it causes the appended text to be placed at the  
10652       beginning of the buffer.

## 10653       **Change Command**

10654       *Synopsis:*       ( . . )c  
10655                       <text>  
10656                       .

10657       The **c** command deletes the addressed lines, then accepts input text that replaces these lines; the  
10658       current line will be set to the address of the last line input; or, if there were none, at the line after  
10659       the last line deleted; if the lines deleted were originally at the end of the buffer, the current line  
10660       number will be set to the address of the new last line; if no lines remain in the buffer, the current  
10661       line number will be set to zero.

## 10662       **Delete Command**

10663       *Synopsis:*       ( . . )d

10664       The **d** command deletes the addressed lines from the buffer. The address of the line after the last  
10665       line deleted will become the current line number; if the lines deleted were originally at the end of  
10666       the buffer, the current line number will be set to the address of the new last line; if no lines  
10667       remain in the buffer, the current line number will be set to zero.

## 10668       **Edit Command**

10669       *Synopsis:*       e [*file*]

10670       The **e** command deletes the entire contents of the buffer and then reads in the file named by the  
10671       pathname *file*. The current line number will be set to the address of the last line of the buffer. If  
10672       no pathname is given, the currently remembered pathname, if any, will be used (see the **f**  
10673       command). The number of bytes read will be written to standard output, unless the **-s** option  
10674       was specified, in the following format:

10675           "%d\\n", <number of bytes read>

10676       The name *file* will be remembered for possible use as a default pathname in subsequent **e**, **E**, **r**  
10677       and **w** commands. If *file* is replaced by **!**, the rest of the line will be taken to be a shell command

10678 line whose output is to be read. Such a shell command line is be remembered as the current *file*.  
 10679 All marks will be discarded upon the completion of a successful **e** command. If the buffer has  
 10680 changed since the last time the entire buffer was written, the user will be warned, as described  
 10681 previously.

## 10682 **Edit Without Checking Command**

10683 *Synopsis:*        **E** [*file*]

10684 The **E** command possesses all properties and restrictions of the **e** command except that the  
 10685 editor will not check to see if any changes have been made to the buffer since the last **w**  
 10686 command.

## 10687 **Filename Command**

10688 *Synopsis:*        **f** [*file*]

10689 If *file* is given, the **f** command will change the currently remembered pathname to *file*; whether  
 10690 the name is changed or not, it then will write the (possibly new) currently remembered  
 10691 pathname to the standard output in the following format:

10692        "%s\n", <pathname>

10693 The current line number is unchanged.

## 10694 **Global Command**

10695 *Synopsis:*        (**l**,**\$**)**g**/*RE/command list*

10696 In the **g** command, the first step is to mark every line that matches the given *RE*. Then, for every  
 10697 such line, the given *command list* will be executed with the current line number set to the address  
 10698 of that line. When the **g** command completes, the current line number will have the value  
 10699 assigned by the last command in the command list. If there were no matching lines, the current  
 10700 line number will not be changed. A single command or the first of a list of commands will  
 10701 appear on the same line as the global command. All lines of a multi-line list except the last line  
 10702 will be ended with a backslash; the **a**, **i** and **c** commands and associated input are permitted. The  
 10703 **.** terminating input mode can be omitted if it would be the last line of the *command list*. An  
 10704 empty *command list* is equivalent to the **p** command. The use of the **g**, **G**, **v**, **V** and **!** commands  
 10705 in the *command list* produces undefined results. Any character other than space or newline can  
 10706 be used instead of a slash to delimit the *RE*. Within the *RE*, the *RE* delimiter itself can be used as  
 10707 a literal character if it is preceded by a backslash.

## 10708 **Interactive Global Command**

10709 *Synopsis:*        (**l**,**\$**)**G**/*RE/*

10710 In the **G** command, the first step is to mark every line that matches the given *RE*. Then, for  
 10711 every such line, that line will be written, the current line number will be set to the address of that  
 10712 line, and any one command (other than one of the **a**, **c**, **i**, **g**, **G**, **v** and **V** commands) can be input  
 10713 and will be executed. A newline character acts as a null command (causing no action to be taken  
 10714 on the current line); an **&** causes the reexecution of the most recent non-null command executed  
 10715 within the current invocation of **G**. Note that the commands input as part of the execution of  
 10716 the **G** command can address and affect any lines in the buffer. The final value of the current line  
 10717 number will be the value set by the last command successfully executed. (Note that the last  
 10718 command successfully executed will be the **G** command itself if a command fails or the null  
 10719 command is specified.) If there were no matching lines, the current line number will not be  
 10720 changed. The **G** command can be terminated by a SIGINT signal. Any character other than

10721 space or newline can be used instead of a slash to delimit the *RE* and the replacement. Within  
 10722 the *RE*, the *RE* delimiter itself can be used as a literal character if it is preceded by a backslash.

### 10723 **Help Command**

10724 *Synopsis:*           h

10725 The **h** command writes a short message to standard output that explains the reason for the most  
 10726 recent ? notification. The current line number is unchanged.

### 10727 **Help-mode Command**

10728 *Synopsis:*           H

10729 The **H** command causes *ed* to enter a mode in which help messages (see the **h** command) will be  
 10730 written to standard output for all subsequent ? notifications. The **H** command alternatively will  
 10731 turn this mode on and off; it is initially off. If the help-mode is being turned on, the **H** command  
 10732 also will explain the previous ? notification, if there was one. The current line number is  
 10733 unchanged.

### 10734 **Insert Command**

10735 *Synopsis:*           ( . ) i  
 10736                       <text>  
 10737                       .

10738 The **i** command inserts the given text before the addressed line; . will be left at the last inserted  
 10739 line or, if there was none, at the addressed line. This command differs from the **a** command only  
 10740 in the placement of the input text. Address 0 is invalid for this command.

### 10741 **Join Command**

10742 *Synopsis:*           ( . , . +1 ) j

10743 The **j** command joins contiguous lines by removing the appropriate newline characters. If  
 10744 exactly one address is given, this command will do nothing. If lines are joined, the current line  
 10745 number will be set to the address of the joined line; otherwise, the current line number is  
 10746 unchanged.

### 10747 **Mark Command**

10748 *Synopsis:*           ( . ) kx

10749 The **m** command marks the addressed line with name *x*, which must be a lower-case letter from  
 10750 the portable character set. The address 'x' then refers to this line; the current line number is  
 10751 unchanged.

### 10752 **List Command**

10753 *Synopsis:*           ( . , . ) l

10754 The **l** command writes to standard output the addressed lines in a visually unambiguous form.  
 10755 The characters listed in the table in the **XBD** specification, **Chapter 3, File Format Notation** (`\`,  
 10756 `\a`, `\b`, `\f`, `\r`, `\t`, `\v`) will be written as the corresponding escape sequence; the `\n` in that table is  
 10757 not applicable. Non-printable characters not in the table will be written as one three-digit octal  
 10758 number (with a preceding backslash character) for each byte in the character (most significant  
 10759 byte first). If the size of a byte on the system is greater than nine bits, the format used for non-  
 10760 printable characters is implementation-dependent.



10761 Long lines will be folded, with the point of folding indicated by writing backslash/newline  
 10762 character; the length at which folding occurs is unspecified, but should be appropriate for the  
 10763 output device. The end of each line will be marked with a "\$". An **l** command can be appended  
 10764 to any other command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w** or **!**. The current line number will be set to the  
 10765 address of the last line written.

#### 10766 **Move Command**

10767 *Synopsis:* ( . , . ) *address*

10768 The **m** command repositions the addressed lines after the line addressed by *address*. Address 0 is  
 10769 valid for *address* and causes the addressed lines to be moved to the beginning of the buffer. It is  
 10770 an error if *address* falls within the range of moved lines. The current line number will be  
 10771 set to the address of the last line moved.

#### 10772 **Number Command**

10773 *Synopsis:* ( . , . ) *n*

10774 The **n** command writes to standard output the addressed lines, preceding each line by its line  
 10775 number and a tab character; the current line number will be set to the address of the last line  
 10776 written. The **n** command can be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w** or **!**.

#### 10777 **Print Command**

10778 *Synopsis:* ( . , . ) *p*

10779 The **p** command writes to standard output the addressed lines; the current line number will be  
 10780 set to the address of the last line written. The **p** command can be appended to any command  
 10781 other than **e**, **E**, **f**, **q**, **Q**, **r**, **w** or **!**.

#### 10782 **Prompt Command**

10783 *Synopsis:* *P*

10784 The **P** command causes *ed* to prompt with an asterisk (\*) (or *string*, if **-p** is specified) for all  
 10785 subsequent commands. The **P** command alternatively turns this mode on and off; it is initially  
 10786 on if the **-p** option is specified, otherwise off. The current line number is unchanged.

#### 10787 **Quit Command**

10788 *Synopsis:* *q*

10789 The **q** command causes *ed* to exit. If the buffer has changed since the last time the entire buffer  
 10790 was written, the user will be warned, as described previously.

#### 10791 **Quit Without Checking Command**

10792 *Synopsis:* *Q*

10793 The **Q** command causes *ed* to exit without checking if changes have been made in the buffer  
 10794 since the last **w** command.

**Read Command**

**Synopsis:**        (*\$*)*r*[*file*]

The *r* command reads in the file named by the pathname *file* and appends it after the addressed line. If no *file* argument is given, the currently remembered pathname, if any, will be used (see *e* and *f* commands). The currently remembered pathname will not be changed unless there is no remembered pathname. Address 0 is valid for *r* and causes the file to be read at the beginning of the buffer. If the read is successful, and *-s* was not specified, the number of bytes read will be written to standard output in the following format:

"%d\n", <number of bytes read>

The current line number will be set to the address of the last line read in. If *file* is replaced by *!*, the rest of the line will be taken to be a shell command line whose output is to be read. Such a shell command line will not be remembered as the current pathname.

**Substitute Command**

**Synopsis:**        (*.*...) *s*/*RE*/*replacement*/*flags*

The *s* command searches each addressed line for an occurrence of the specified *RE* and replace either the first or all (non-overlapped) matched strings with the *replacement*; see the following description of the *g* suffix. It is an error if the substitution fails on every addressed line. Any character other than space or newline can be used instead of a slash to delimit the *RE* and the replacement. Within the *RE*, the *RE* delimiter itself can be used as a literal character if it is preceded by a backslash. The current line will be set to the address of the last line on which a substitution occurred.

An ampersand (&) appearing in the *replacement* will be replaced by the string matching the *RE* on the current line. The special meaning of "&" in this context can be suppressed by preceding it by backslash. As a more general feature, the characters \n, where *n* is a digit, will be replaced by the text matched by the corresponding back-reference expression. When the character "%" is the only character in the *replacement*, the *replacement* used in the most recent substitute command will be used as the *replacement* in the current substitute command; if there was no previous substitute command, the use of "%" in this manner is an error. The "%" loses its special meaning when it is in a replacement string of more than one character or is preceded by a backslash. For each backslash (\) encountered in scanning *replacement* from beginning to end, the following character loses its special meaning (if any). It is unspecified what special meaning is given to any character other than "&", "\%", or digits.

A line can be split by substituting a newline character into it. The application must escape the newline character in the *replacement* by preceding it by backslash. Such substitution cannot be done as part of a *g* or *v* command list. The current line number will be set to the address of the last line on which a substitution is performed. If no substitution is performed, the current line number is unchanged. If a line is split, a substitution is considered to have been performed on each of the new lines for the purpose of determining the new current line number. A substitution is considered to have been performed even if the replacement string is identical to the string that it replaces.

The value of *flags* must be zero or more of:

*count*   Substitute for the *count*th occurrence only of the *RE* found on each addressed line.

*g*        Globally substitute for all non-overlapping instances of the *RE* rather than just the first one. If both *g* and *count* are specified, the results are unspecified.

10839 **l** Write to standard output the final line in which a substitution was made. The line will  
10840 be written in the format specified for the **l** command.

10841 **n** Write to standard output the final line in which a substitution was made. The line will  
10842 be written in the format specified for the **n** command.

10843 **p** Write to standard output the final line in which a substitution was made. The line will  
10844 be written in the format specified for the **p** command.

### 10845 **Copy Command**

10846 *Synopsis:* `(... )taddress`

10847 The **t** command is equivalent to the **m** command, except that a copy of the addressed lines will  
10848 be placed after address *address* (which can be 0); the current line number will be set to the  
10849 address of the last line added.

### 10850 **Undo Command**

10851 *Synopsis:* `u`

10852 The **u** command nullifies the effect of the most recent command that modified anything in the  
10853 buffer, namely the most recent **a**, **c**, **d**, **g**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, **G** or **V** command. All changes made  
10854 to the buffer by a **g**, **G**, **v** or **V** global command will be undone as a single change; if no changes  
10855 were made by the global command (such as with **g/RE/p**), the **u** command will have no effect.  
10856 The current line number will be set to the value it had immediately before the command being  
10857 undone started.

### 10858 **Global Non-matched Command**

10859 *Synopsis:* `(1,$)v/RE/command list`

10860 This command is equivalent to the global command **g** except that the lines that are marked  
10861 during the first step will be those that do not match the *RE*.

### 10862 **Interactive Global Not-matched Command**

10863 *Synopsis:* `(1,$)V/RE/`

10864 This command is equivalent to the interactive global command **G** except that the lines that are  
10865 marked during the first step will be those that do not match the *RE*.

### 10866 **Write Command**

10867 *Synopsis:* `(1,$)w[file]`

10868 The **w** command writes the addressed lines into the file named by the pathname *file*. The  
10869 command will create the file, if it does not exist, or will replace the contents of the existing file.  
10870 The currently remembered pathname will not be changed unless there is no remembered  
10871 pathname. If no pathname is given, the currently remembered pathname, if any, will be used  
10872 (see the **e** and **f** commands); the current line number is unchanged. If the command is  
10873 successful, the number of bytes written will be written to standard output, unless the **-s** option  
10874 was specified, in the following format:

10875 `"%d\n", <number of bytes written>`

10876 If *file* begins with "!", the rest of the line will be taken to be a shell command line whose standard  
10877 input will be the addressed lines. Such a shell command line will not be remembered as the  
10878 current pathname. This usage of the write command with "!" will not be considered as a "last w

10879 command that wrote the entire buffer'', as described previously; thus, this alone will not prevent  
 10880 the warning to the user if an attempt is made to destroy the editor buffer via the **e** or **q**  
 10881 commands.

## 10882 **Line Number Command**

10883 *Synopsis:*           ( *\$* ) =

10884 The line number of the addressed line will be written to standard output in the following format:

10885       "%d\n", <*line number*>

10886 The current line number is unchanged by this command.

## 10887 **Shell Escape Command**

10888 *Synopsis:*           ! *command*

10889 The remainder of the line after the **!** will be sent to the command interpreter to be interpreted as  
 10890 a shell command line. Within the text of that shell command line, the unescaped character **%**  
 10891 will be replaced with the remembered pathname; if a **!** appears as the first character of the  
 10892 command, it will be replaced with the text of the previous shell command executed via **!**. Thus,  
 10893 **!!** will repeat the previous *!command*. If any replacements of **%** or **!** are performed, the modified  
 10894 line will be written to the standard output before *command* is executed. The **!** command will  
 10895 write:

10896       "! \n"

10897 to standard output upon completion, unless the **-s** option is specified. The current line number  
 10898 is unchanged.

## 10899 **Null Command**

10900 *Synopsis:*           ( . +1 )

10901 An address alone on a line causes the addressed line to be written. A newline character alone is  
 10902 equivalent to **.+1p**. The current line number will be set to the address of the written line.

## 10903 **EXIT STATUS**

10904 The following exit values are returned:

- 10905       0   Successful completion without any file or command errors.
- 10906       >0   An error occurred.

## 10907 **CONSEQUENCES OF ERRORS**

10908 When an error in the input script is encountered, or when an error is detected that is a  
 10909 consequence of the data (not) present in the file or due to an external condition such as a read or  
 10910 write error:

- 10911       • If the standard input is a terminal device file, all input will be flushed, and a new command  
 10912       read.
- 10913       • If the standard input is a regular file, *ed* will terminate with a non-zero exit status.

## 10914 **APPLICATION USAGE**

10915 Because of the extremely terse nature of the default error messages, the prudent script writer  
 10916 will begin the *ed* input commands with an **H** command, so that if any errors do occur at least  
 10917 some clue as to the cause will be made available.

10918 **EXAMPLES**

10919       None.

10920 **FUTURE DIRECTIONS**

10921       The obsolescent single-minus form may be withdrawn in a future issue. Applications should  
10922       use the **–s** option.

10923       The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
10924       interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
10925       corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
10926       finalised.

10927 **SEE ALSO**10928       *ex, sed, sh, vi.*10929 **CHANGE HISTORY**

10930       First released in Issue 2.

10931 **Issue 4**

10932       Aligned with the ISO/IEC 9945-2: 1993 standard.

10933 **Issue 5**10934       In the OPTIONS section, the meaning of **–s** and **"–"** is clarified.

10935       Second FUTURE DIRECTION added.

10936 **NAME**10937 **egrep** — search a file with an ERE pattern (**LEGACY**)10938 **SYNOPSIS**10939 OB `egrep [ -c | -l ][-inv] -e pattern_list [file...]`10940 OB `egrep [ -c | -l ][-inv] -f pattern_list [file...]`10941 OB `egrep [ -c | -l ][-inv] pattern_list [file...]`10942 **DESCRIPTION**10943 The name *egrep* is an obsolescent version equivalent to *grep -E*.10944 A command invoking the *egrep* utility with the *-e* option specified is equivalent to the  
10945 command:10946 `grep -E [ -c | -l ][-inv] -e pattern_list [file...]`10947 A command invoking the *egrep* utility with the *-f* option specified is equivalent to the command:10948 `grep -E [ -c | -l ][-inv] -f pattern_list [file...]`10949 A command invoking the *egrep* utility with neither the *-e* nor the *-f* option specified is  
10950 equivalent to the command:10951 `grep -E [ -c | -l ][-inv] pattern_list [file...]`10952 **OPTIONS**10953 Refer to *grep*.10954 **OPERANDS**10955 Refer to *grep*.10956 **STDIN**10957 Refer to *grep*.10958 **INPUT FILES**10959 Refer to *grep*.10960 **ENVIRONMENT VARIABLES**10961 Refer to *grep*.10962 **ASYNCHRONOUS EVENTS**10963 Refer to *grep*.10964 **STDOUT**10965 Refer to *grep*.10966 **STDERR**10967 Refer to *grep*.10968 **OUTPUT FILES**10969 Refer to *grep*.10970 **EXTENDED DESCRIPTION**10971 Refer to *grep*.10972 **EXIT STATUS**10973 Refer to *grep*.10974 **CONSEQUENCES OF ERRORS**10975 Refer to *grep*.

10976 **APPLICATION USAGE**

10977 Unlike *grep -E*, multiple *-e* or *-f* options produce undefined results. Adjacent newline  
10978 characters in the *pattern* operand or *-e pattern\_list* option-argument also produce undefined  
10979 results.

10980 Applications should migrate to the *grep -E* command.

10981 **EXAMPLES**

10982 Refer to *grep*.

10983 **FUTURE DIRECTIONS**

10984 Refer to *grep*.

10985 **SEE ALSO**

10986 *grep*.

10987 **CHANGE HISTORY**

10988 First released in Issue 2.

10989 **Issue 4**

10990 Aligned with the ISO/IEC 9945-2: 1993 standard.

10991 Separated from the *fgrep* description.

10992 **Issue 5**

10993 Marked LEGACY.

## 10994 NAME

10995       env — set the environment for command invocation

## 10996 SYNOPSIS

10997       env [-i][name=value]... [utility [argument...]]

10998 OB     env [-][name=value]... [utility [argument...]]

## 10999 DESCRIPTION

11000       The *env* utility will obtain the current environment, modify it according to its arguments, then  
 11001       invoke the utility named by the *utility* operand with the modified environment.

11002       Optional arguments will be passed to *utility*.

11003       If no *utility* operand is specified, the resulting environment will be written to the standard  
 11004       output, with one *name=value* pair per line.

## 11005 OPTIONS

11006 OB     The *env* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 11007       for its non-standard usage of "-".

11008       The following options are supported:

11009       -i

11010 OB     -

11011       Invoke *utility* with exactly the environment specified by the arguments; the inherited  
 11012       environment will be ignored completely.

## 11013 OPERANDS

11014       The following operands are supported:

11015       *name=value*

11016       Arguments of the form *name=value* modify the execution environment, and are placed  
 11017       into the inherited environment before the *utility* is invoked.

11018       *utility*   The name of the utility to be invoked. If the *utility* operand names any of the special  
 11019       built-in utilities in Section 2.14 on page 67, the results are undefined.

11020       *argument*

11021       A string to pass as an argument for the invoked utility.

## 11022 STDIN

11023       Not used.

## 11024 INPUT FILES

11025       None.

## 11026 ENVIRONMENT VARIABLES

11027       The following environment variables affect the execution of *env*:

11028       *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 11029       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 11030       default locale will be used. If any of the internationalisation variables contains an  
 11031       invalid setting, the utility will behave as if none of the variables had been defined.

11032       *LC\_ALL*

11033       If set to a non-empty string value, override the values of all the other  
 11034       internationalisation variables.

11035       *LC\_CTYPE*

11036       Determine the locale for the interpretation of sequences of bytes of text data as  
 11037       characters (for example, single- as opposed to multi-byte characters in arguments).



11038 **LC\_MESSAGES**  
 11039 Determine the locale that should be used to affect the format and contents of diagnostic  
 11040 messages written to standard error.

11041 EX **NLSPATH**  
 11042 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

11043 **PATH** Determine the location of the *utility*, as described in the **XBD** specification, **Chapter 6**,  
 11044 **Environment Variables**. If *PATH* is specified as a *name=value* operand to *env*, the *value*  
 11045 given will be used in the search for *utility*.

11046 **ASYNCHRONOUS EVENTS**  
 11047 Default.

11048 **STDOUT**  
 11049 If no *utility* operand is specified, each *name=value* pair in the resulting environment will be  
 11050 written in the form:  
 11051 "%s=%s\n", <name>, <value>  
 11052 If the *utility* operand is specified, the *env* utility will not write to standard output.

11053 **STDERR**  
 11054 Used only for diagnostic messages.

11055 **OUTPUT FILES**  
 11056 None.

11057 **EXTENDED DESCRIPTION**  
 11058 None.

11059 **EXIT STATUS**  
 11060 If the *utility* utility is invoked, the exit status of *env* will be the exit status of *utility*; otherwise, the  
 11061 *env* utility will exit with one of the following values:

|       |       |                                                                             |
|-------|-------|-----------------------------------------------------------------------------|
| 11062 | 0     | The <i>env</i> utility completed successfully.                              |
| 11063 | 1-125 | An error occurred in the <i>env</i> utility.                                |
| 11064 | 126   | The utility specified by <i>utility</i> was found but could not be invoked. |
| 11065 | 127   | The utility specified by <i>utility</i> could not be found.                 |

11066 **CONSEQUENCES OF ERRORS**  
 11067 Default.

11068 **APPLICATION USAGE**  
 11069 The *command*, *env*, *nice*, *nohup*, *time* and *xargs* utilities have been specified to use exit code 127 if  
 11070 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
 11071 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
 11072 used for other meanings; most utilities use small values for “normal error conditions” and the  
 11073 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 11074 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
 11075 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 11076 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 11077 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
 11078 any other reason.

11079 Historical implementations of the *env* utility use the **XSH** specification *execvp()* or *execlp()*  
11080 functions to invoke the specified utility; this provides better performance and keeps users from  
11081 having to escape characters with special meaning to the shell. Therefore, shell functions, special  
11082 built-ins and built-ins that are only provided by the shell are not found.

#### 11083 **EXAMPLES**

11084 The following command:

11085 `env -i PATH=/mybin mygrep xyz myfile`

11086 invokes the command *mygrep* with a new *PATH* value as the only entry in its environment. In  
11087 this case, *PATH* is used to locate *mygrep*, which then must reside in **/mybin**.

#### 11088 **FUTURE DIRECTIONS**

11089 None.

#### 11090 **SEE ALSO**

11091 Section 2.5 on page 27.

#### 11092 **CHANGE HISTORY**

11093 First released in Issue 2.

#### 11094 **Issue 4**

11095 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 11096 NAME

11097 ex — text editor

## 11098 SYNOPSIS

11099 EX ex [-rR][-l][-s | -v ][-c *command*]-t *tagstring*[-w *size*][*file...*]11100 OB ex [-rR][-l][- | -v ][+*command*][-t *tagstring*][-w *size*][*file...*]

## 11101 DESCRIPTION

11102 The ex utility is a line-oriented text editor that supports both line and full-screen editing (see *vi*).

11103 Certain block-mode terminals do not have all the capabilities necessary to support the complete  
 11104 ex definition, such as the full-screen editing commands (*visual* mode) or *open* mode. When these  
 11105 commands cannot be supported on such terminals, this condition will not produce an error  
 11106 message such as “not an editor command” nor report a syntax error. The implementation may  
 11107 either accept the commands and produce results on the screen that are the result of an  
 11108 unsuccessful attempt to meet the requirements of this specification or report an error describing  
 11109 the terminal-related deficiency. The affected commands are noted as they occur later in this  
 11110 section.

## 11111 OPTIONS

11112 The ex utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

11113 The following options are supported:

11114 -c *command*11115 OB +*command*

11116 Begin editing by executing the specified ex command-mode commands. As with  
 11117 normal editing command-line entries, the *command* option-argument can consist of  
 11118 multiple ex commands separated by vertical-line characters (|). The use of commands  
 11119 that enter input or visual modes in this manner produces undefined results.

11120 EX -l (The letter ell.) Set lisp mode; indents appropriately for Lisp code; the {}, [], [[ and ]]  
 11121 commands in visual mode are modified to have meaning for Lisp.

11122 -r Recover the named files after an editor or system crash, after the editor has been  
 11123 terminated by a signal, or after the use of a **preserve** editor command. A *crash* in this  
 11124 context is an unexpected failure of the system or utility that requires restarting the  
 11125 failed system or utility. A system crash implies that any utilities running at the time  
 11126 also crash. In the case of an editor or system crash, the degree of recovery (the number  
 11127 of changes to the buffer since the most recent **preserve** command) available is  
 11128 unspecified.

11129 If no *file* operands are given, all other options, the *EXINIT* variable and any .**exrc** files  
 11130 will be ignored; a list of all recoverable files available to the invoking user will be  
 11131 written; and ex will exit without reading files or processing user commands.

11132 -R Set read-only mode, preventing accidental overwriting of the files. Any command that  
 11133 would write to a file will require the "!" suffix (see, for example, the **write** command) to  
 11134 be effective in this mode.

11135 -s

11136 OB - Prepare ex for batch use by taking the following actions:

11137 • Suppress writing prompts and informational (but not diagnostic) messages.

- 11138 • Ignore the value of *TERM* and any implementation default terminal type and
- 11139 assume the terminal is a type incapable of supporting visual mode; see the **visual**
- 11140 command and the description of *vi*.
- 11141 • Suppress the use of the *EXINIT* environment variable and the reading of any **.exrc**
- 11142 file (see the EXTENDED DESCRIPTION section).

11143 **-t tagstring**  
 11144 Edit the file containing the specified *tagstring* and proceed as if the first command were  
 11145 **:tag tagstring**. (See *ctags*.) The tags feature represented by **-t tagstring** and the **ta**  
 11146 command are optional. It is provided on any system that also provides a conforming  
 11147 implementation of *ctags*; otherwise, the use of **-t** produces undefined results.

11148 **-v** Invoke *vi*.

11149 **-w size**  
 11150 Set the value of the *window* editor option to *size*.

11151 OB If both the **-t tagstring** and **-c command** (or the obsolescent **+command**) options are given, the  
 11152 **-t tagstring** will be processed first; that is, the file containing the tag is selected by **-t** and then  
 11153 the command is executed.

## 11154 OPERANDS

11155 The following operand is supported:

11156 **file** A pathname of a file to be edited.

## 11157 STDIN

11158 The standard input must be a text file consisting of commands, as described in the EXTENDED  
 11159 DESCRIPTION section.

## 11160 INPUT FILES

11161 Input files must be text files or files that would be text files except for an incomplete last line that  
 11162 is not longer than {LINE\_MAX}-1 bytes in length and contains no NUL characters. The editing  
 11163 of other forms of files may optionally be allowed by *ex* implementations.

11164 The **.exrc** files (see the EXTENDED DESCRIPTION section) must be text files consisting of  
 11165 commands.

11166 By default, *ex* reads lines from the files to be edited without interpreting any of those lines as any  
 11167 form of editor command.

## 11168 ENVIRONMENT VARIABLES

11169 The following environment variables affect the execution of *ex*:

### 11170 COLUMNS

11171 Override the system-selected horizontal screen size. See the **XBD** specification,  
 11172 **Chapter 6, Environment Variables** for valid values and results when it is unset or null.

11173 **EXINIT** Determine a list of *ex* commands that are executed on editor start-up, before reading  
 11174 the first file. The list can contain multiple commands by separating them using a  
 11175 vertical-line (|) character. See the EXTENDED DESCRIPTION section for more details  
 11176 of the initialisation phase.

11177 **HOME** Determine a pathname of a directory that will be searched for an editor start-up file  
 11178 named **.exrc**; see the EXTENDED DESCRIPTION section for details.

11179 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 11180 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 11181 default locale will be used. If any of the internationalisation variables contains an

- 11182               invalid setting, the utility will behave as if none of the variables had been defined.
- 11183       *LC\_ALL*
- 11184               If set to a non-empty string value, override the values of all the other
- 11185               internationalisation variables.
- 11186       *LC\_COLLATE*
- 11187               Determine the locale for the behaviour of ranges, equivalence classes and multi-
- 11188               character collating elements within regular expressions.
- 11189       *LC\_CTYPE*
- 11190               Determine the locale for the interpretation of sequences of bytes of text data as
- 11191               characters (for example, single- as opposed to multi-byte characters in arguments and
- 11192               input files), the behaviour of character classes within regular expressions, the
- 11193               classification of characters as upper- or lower-case letters, the case conversion of letters,
- 11194               and the detection of word boundaries.
- 11195       *LC\_MESSAGES*
- 11196               Determine the locale that should be used to affect the format and contents of diagnostic
- 11197               messages written to standard error.
- 11198       *LINES*   Override the system-selected vertical screen size, used as the number of lines in a
- 11199               screenful and the vertical screen size in visual mode. See the **XBD** specification,
- 11200               **Chapter 6, Environment Variables** for valid values and results when it is unset or null.
- 11201 EX       *NLSPATH*
- 11202               Determine the location of message catalogues for the processing of *LC\_MESSAGES*.
- 11203       *PATH*    Determine the search path for the shell command specified in the editor commands
- 11204               **shell**, **read** and **write**; see the description of command search and execution in
- 11205               **Command Search and Execution** on page 47.
- 11206       *SHELL*   Determine the preferred command-line interpreter for use in **!**, **shell**, **read** and other
- 11207               commands with an operand of the form **!string**. For the **shell** command, the program
- 11208               will be invoked with the single argument **-i**, for all others it will be invoked with the
- 11209               two arguments **-c** and *string*. If no *SHELL* environment variable is set, or it is set to a
- 11210               null string, the *sh* utility will be used.
- 11211       *TERM*    Determine the name of the terminal type. If this variable is unset or null, an
- 11212               unspecified default terminal type will be used.
- 11213 **ASYNCHRONOUS EVENTS**
- 11214       The following actions will be taken upon receipt of signals:
- 11215       **SIGINT**   When an interrupt occurs, *ex* will alert the terminal and write a message. The current
- 11216               editor command will be aborted and *ex* will return to the command level and prompt
- 11217               for another command. If the standard input is not a terminal device, *ex* will exit at the
- 11218               interrupt and return a non-zero exit status. (The alerting action can be modified by the
- 11219               use of the **errorbells** editor option; see **Edit Options in ex** on page 325.)
- 11220       **SIGCONT**
- 11221               The screen will be refreshed (if in visual mode).
- 11222       **SIGHUP**
- 11223               If the current buffer has changed since the last **e** or **w** command, *ex* will attempt to save
- 11224               the current file in a state such that it can be recovered later by an *ex -r* command.
- 11225       The action taken for all other signals is unspecified.

11226 **STDOUT**

11227       The standard output is used only for writing prompts to the user, for informational messages  
11228       and for writing lines from the file.

11229 **STDERR**

11230       Used only for diagnostic messages.

11231 **OUTPUT FILES**

11232       The output from *ex* must be text files that are identical to the input files if no changes have been  
11233       made to the files by commands, with the exception that in all cases where a forced session  
11234       termination (the *ex* command *q!*) has not been issued prior to any file write, a trailing newline  
11235       character will be added to the last line of the file if one was not present in the input.

11236 **EXTENDED DESCRIPTION**

11237       The pathname of the file being edited by *ex* is referred to as the *current* file. The text of the file is  
11238 *EX*       read into a working version of the file (called *buffer* in this section; intermediate versions of the  
11239       buffer may be kept in a file that is created in the directory indicated by the **directory** editor  
11240       option) and all editing changes are performed on that version; the changes have no effect on the  
11241       original file until an *ex* command causes the file to be written out. Lines in the buffer may each  
11242       be limited to {LINE\_MAX} bytes and an error message may be written if the limit is exceeded  
11243       during editing.

11244       The *alternative* pathname is the name of the last file mentioned in an editor command, or the  
11245       previous current pathname if the last file mentioned became the current file. When the character  
11246       "%" appears in a pathname entered as part of a command argument, it is replaced by the current  
11247       pathname; the character "#" is replaced by the alternative pathname. Any character, including  
11248       "%" and "#", retains its literal value (is escaped) when preceded by a backslash.

11249       When an error occurs, *ex* will alert the terminal and write a message. (The alerting action can be  
11250       modified by the use of the **errorbells** editor option.)

11251       If the system crashes, *ex* will attempt to preserve the buffer if any unwritten changes were made.  
11252       The command-line option **-r** can be used to retrieve the saved changes.

11253       During initialisation, before the first file is read or any user commands from the terminal are  
11254       processed, if the environment variable *EXINIT* is set, the editor will execute the *ex* commands  
11255       contained in that variable. If the variable is not set, *ex* will attempt to read commands from the  
11256       file *\$HOME/.exrc* (the file *.exrc* in the directory referred to by the *HOME* environment variable).  
11257       If and only if *EXINIT* or *\$HOME/.exrc* sets the editor option **exrc**, *ex* finally will attempt to read  
11258       commands from a file *.exrc* in the current directory. In the event that *EXINIT* is not set and the  
11259       current directory is the user's home directory, any *.exrc* file will only be processed once. No  
11260       *.exrc* file will be read unless it is owned by the same user ID as the effective user ID of the  
11261       process. After any *.exrc* files are processed, any commands specified by the **-c** option will be  
11262       processed.

11263       By default, *ex* starts in the command mode, which is indicated by the **:** prompt. The input  
11264       mode can be entered by **append**, **insert** or **change** commands; it can be exited (and command  
11265       mode reentered) by typing a period (.) alone at the beginning of a line. There is one other mode,  
11266       visual mode, in which full-screen editing is available. This is described more fully under the  
11267       **visual** command and in the *vi* utility description. The command line can consist of multiple *ex*  
11268       commands separated by vertical-line characters (|). The use of commands that enter input or  
11269       visual modes in this manner, unless they are the final command on the line, produces undefined  
11270       results.

11271       Command lines beginning with the double-quote character (") are ignored. This can be used for  
11272       comments in an editor script.

11273 **Addressing in ex**

11274 Addressing in *ex* relates to the *current line*. In general, the current line is the last line affected by a  
 11275 command; the exact effect on the current line is discussed under the description of each  
 11276 command. When the buffer contains no lines, the current line is set to zero.

11277 Addresses are constructed by one of the following methods:

- 11278 1. The address `.` (period) refers to the current line.
- 11279 2. The address `"$"` refers to the last line of the buffer.
- 11280 3. The address `n`, where `n` is a decimal number, refers to the `n`th line of the buffer.
- 11281 4. The address `'x` refers to the line marked with the mark-name character `x`, which must be a  
 11282 lower-case letter of the POSIX locale. Lines can be marked with the **ma** or **k** commands  
 11283 described below.
- 11284 5. A regular expression (RE) enclosed by slashes (`/`) is an address, and refers to the first line  
 11285 found by searching forward from the line following the current line toward the end of the  
 11286 buffer and stopping at the first line containing a string matching the RE. The second slash  
 11287 can be omitted at the end of a command line. If the **wrapscan** option is set, the search will  
 11288 wrap around to the beginning of the buffer and continue up to and including the current  
 11289 line, so that the entire buffer is searched.
- 11290 6. An RE enclosed in question marks (`?`) addresses the first line found by searching backward  
 11291 from the line preceding the current line toward the beginning of the buffer and stopping at  
 11292 the first line containing a string matching the RE. The second question mark can be  
 11293 omitted at the end of a command line. If the **wrapscan** option is set, the search will wrap  
 11294 around from the beginning of the buffer to the end of the buffer and continue up to and  
 11295 including the current line, so that the entire buffer is searched.
- 11296 7. An address followed by a plus sign (`+`) or a minus sign (`-`) followed by a decimal number  
 11297 is an offset address, and refers to the first address plus (respectively minus) the indicated  
 11298 number of lines. If the address is omitted, the addition or subtraction is taken with respect  
 11299 to the current line.
- 11300 8. An address of `"+"` or `"-"` followed by a number is taken with respect to the current line  
 11301 number; for example, `-5` is understood to mean `.-5`.
- 11302 9. An address ending with `"+"` or `"-"` has 1 added to or subtracted from the address,  
 11303 respectively. As a consequence of this rule and of rule 8 above, the address `"-"` refers to the  
 11304 line preceding the current line. Moreover, trailing `"+"` and `"-"` characters have a cumulative  
 11305 effect; for example, `--` refers to the current line less 2.
- 11306 10. A percent sign (`%`) stands for the address pair `1,$`.

11307 Commands require zero, one or two addresses. See the descriptions of *line* and *range* in  
 11308 **Command Descriptions in ex** on page 312. Commands that require zero addresses regard the  
 11309 presence of an address as an error.

11310 Adjacent addresses in a *range* must be separated from each other by a comma (`,`) or a semicolon  
 11311 (`;`). In the latter case, the current line (`.`) is set to the first address, and only then is the second  
 11312 address calculated. This feature can be used to determine the starting line for forward and  
 11313 backward searches (see rules 5 and 6 above). The second address of any two-address sequence  
 11314 corresponds to a line that follows, in the buffer, the line corresponding to the first address. The  
 11315 first address must be less than or equal to the second address. The first address must be greater  
 11316 than or equal to the first line of the editing buffer and the last address must be less than or equal  
 11317 to the last line of the editing buffer. Any other case is an error.

11318 All of the following examples are valid *addresses*:

11319     +++     three lines after the current line  
 11320     /re/—    one line before the next occurrence of *re*  
 11321     -2        two lines before the current line.

## 11322 Command Descriptions in ex

11323 The following symbols are used in this section to represent optional modifiers. Any or all can be  
 11324 omitted; the defaults are shown.

11325     *line*     A single line address, given in any of the forms described in **Addressing in ex** on page  
 11326                311; the default for *line* is the current line.

11327     *range*    A *line*, or a pair of line addresses, separated by a comma or semicolon (see **Addressing**  
 11328                **in ex** on page 311 for the difference between the two); the default for *range* is the  
 11329                current line only (.,.). A percent sign (%) stands for the range (1,\$). If the range  
 11330                specified is such that the starting address exceeds the ending address, the range is  
 11331                invalid and the command will not be performed. If more than the expected number of  
 11332                addresses are given in a range, the greatest valid number of the last ones given will be  
 11333                used. For example, 1,3,5p prints lines 3 to 5, inclusive (because two is the greatest valid  
 11334                number in the range accepted by **print**).

11335     *count*    A positive integer, specifying the number of lines to be affected by the command; the  
 11336                default for *count* is 1.

11337     *flags*     One of the characters "#", p or l (ell), or both "#" and l to add numbers to list-format  
 11338                output. When a command with such a flag completes, the addressed lines will be  
 11339                written out as if by the corresponding #, p or l command. The use of *flags* applies to all  
 11340                lines written by the **list**, **number**, **open**, **print**, **substitute**, **visual**, **&** and **z** commands;  
 11341                for other commands, it applies to the current line at the completion of the command. In  
 11342                addition, any number of "+" or "-" characters can also be given after the flags, in which  
 11343                case the line written is not the one affected by the command, but rather the line  
 11344                addressed by the offset address as described above. The default for *flags* is null.

11345     *buffer*    One of a number of named areas for saving text. The named buffers are specified by  
 11346                the lower-case letters of the POSIX locale. Specifying *buffer* causes the area of text  
 11347                affected by the command to be stored into the buffer as it was before the command  
 11348                took effect. This argument is also used on the **put** command and the visual mode put  
 11349                commands (**p** and **P**) to specify the buffer that will provide the text to insert.

11350                If the buffer name is specified in upper-case, and the buffer is to be modified (as with a  
 11351                deletion or yanking command), the buffer will be appended to rather than being  
 11352                overwritten. If the buffer is not to be modified (as in a visual mode **put** command) the  
 11353                buffer name can be specified in lower-case or upper-case with the same results. There  
 11354                is also one unnamed buffer, which is the repository for all text deleted (with the **delete**  
 11355                or visual mode **d** command) or yanked (with the **yank** or visual mode **y** command)  
 11356                when no buffer is specified.

11357                There are also numbered buffers, 1 to 9, inclusive, which are accessible only from visual  
 11358                mode. These buffers are special in that, in visual mode, when deleted text is placed in  
 11359                the unnamed buffer, it also is placed in buffer 1, the previous contents of buffer 1 are  
 11360                placed in buffer 2, and so on. Any text in buffer 9 will be lost. Text that is yanked (or  
 11361                otherwise copied) into the unnamed buffer does not modify the numbered buffers.  
 11362                Text cannot be placed directly into the numbered buffers although it can be retrieved  
 11363                from them by using a visual mode **put** command with the buffer name given as a



11364 number. When the *buffer* modifier is not used in the commands below, the unnamed  
 11365 buffer is the default.

11366 *file* A pattern used to derive a pathname; the default is the current file, as defined above. If  
 11367 no current file has yet been established, a warning will be written and the command  
 11368 will be aborted, except where specifically noted in the individual command  
 11369 descriptions that follow. The pattern will be subjected to the process of shell word  
 11370 expansions (see Section 2.6 on page 31); if more than a single pathname results and the  
 11371 command is expecting one file, the effects are unspecified.

11372 *word* In the POSIX locale, a *word* consists of a maximal sequence of letters, digits and  
 11373 underscores, delimited at both ends by characters other than letters, digits or  
 11374 underscores, or by the beginning or end of a line or the file.

11375 **!** A character that can be appended to the command to modify its operation, as detailed  
 11376 in the individual command descriptions.

11377 If both a *count* and a *range* are specified for a command that uses them, the number of lines  
 11378 affected will be taken from the *count* value rather than the *range*. The starting line for the  
 11379 command is taken to be the first line addressed by the *range*.

11380 When only a *line* or *range* is specified with no command, the implied command is either a **print**,  
 11381 **list** or **number** (**p**, **l** or **#**). The command selected will be the last of these three commands to be  
 11382 used, including use as a *flag*. When no range or count is specified and the command line is a  
 11383 blank line, the current line will be written, and the current line will be set to **+.1**.

11384 Zero or more blank characters can precede or follow the addresses, *count*, *flags* or command  
 11385 name. Any object following a command name (such as *buffer*, *file* and so on) that begins with an  
 11386 alphabetic character will be separated from the command name with at least one blank  
 11387 character.

11388 For each of the commands listed below, the command can be entered as the abbreviation (those  
 11389 characters in the Synopsis command word preceding the **]**), the full command (all characters  
 11390 shown for the command word, omitting the **[** and **]**), or any subset of the characters of the full  
 11391 command down to the abbreviation. For example, the **args** command (shown as **ar[gs]** in the  
 11392 Synopsis) can be entered as **ar**, **arg** or **args**.

11393 **Abbreviate**

11394 *Synopsis:*    **ab**[brev] *word rhs*

11395 Add the named abbreviation to the current abbreviation list. In visual mode, if *word* is typed so  
 11396 that it is preceded and followed by characters that cannot be part of a *word* (as defined  
 11397 previously), it will be replaced by the string *rhs*.

11398 **Append**

11399 *Synopsis:*    **[line]** **a**[ppend][**!**]

11400 Enter input mode; the input text will be placed after the specified line. If line 0 is specified, the  
 11401 text will be placed at the beginning of the buffer. The current line indicator will be set to the last  
 11402 input line; if no lines are input, it will be set to the target line, or to the first line of the file if a  
 11403 target of 0 was specified. Following the command name with **!** causes the **autoindent** editor  
 11404 option setting to be toggled for the duration of this command only.

**Arguments**

*Synopsis:*     ar[gs]

Write the argument list with the current argument inside "[" and "]". The argument list is the list of operands on start-up, which can subsequently be replaced by the operands of the **next** command.

**Change**

*Synopsis:*     [range] c[hange][!] [count]

Enter input mode; the input text will replace the specified lines (*range*). The current line indicator will be set to the last line input; if no lines were input, it will be set to the line before the target line, or to the first line of the file if there are no lines preceding the target. Following the command name with "!" causes the **autoindent** editor option setting to be toggled for the duration of this command only.

**Change Directory**

*Synopsis:*     chd[ir][!] [directory]

*Synopsis:*     cd[!] [directory]

Change the current working directory to *directory*. The argument will be subjected to the process of shell word expansions (see Section 2.6 on page 31). When invoked with no *directory* argument, and the *HOME* environment variable is set to a non-empty value, the directory name in the *HOME* environment variable will become the new working directory. If *HOME* is empty or undefined, the default behaviour is implementation-dependent. If the current buffer has been modified since the last write, a warning will be written and the command will fail. This warning can be overridden by appending "!" to the command name, which will allow the command to complete.

**Copy**

*Synopsis:*     [range] co[py] line [flags]

*Synopsis:*     [range] t line [flags]

Place a copy of the specified lines (*range*) after the specified destination *line*; line 0 specifies that the lines will be placed at the beginning of the buffer.

**Delete**

*Synopsis:*     [range] d[ele]te [buffer] [count] [flags]

Delete the specified lines from the buffer. If a named *buffer* is specified, the deleted text will be saved in it; otherwise, the deleted text will be saved in the unnamed buffer. If the command name is followed by a letter that could be interpreted as either a buffer name or a *flag* value (because neither a *count* nor an additional *flags* value was given), **ex** will consider the letter to be a *flags* value if the letter directly follows the command name, without any blank character separation; if the letter is preceded by one of more blank characters, it is considered a buffer name.

For example:

**1dp** or **1deletep**     Deletes the first line and prints the line that was second.

**1d p**                 Deletes the first line, saving it in buffer p.

11445       **1d p1l**           (Pee-one-ell.) Deletes the first line, saving it in buffer p, and listing the line  
 11446                           that was second.

11447       The current line indicator will be set to the line following the deleted lines, or to the last line if  
 11448       the deleted lines were at the end.

11449       **Edit**

11450       *Synopsis:*     e[dit][!] [+line] [file]  
 11451       *Synopsis:*     ex[!] [+line] [file]

11452       Begin editing *file*. If the current buffer has been modified since the last write, a warning will be  
 11453       written and the command will be aborted. This action can be overridden by appending the  
 11454       character "!" to the command name (e! *file*). The current line indicator will be affected as  
 11455       follows:

- 11456       • If *file* is omitted or results in the current file, the current line indicator will not be changed.
- 11457       • Otherwise, the current line indicator will be the last line of the buffer; however, if this  
 11458       command is executed from within visual mode, the current line indicator will be the first line  
 11459       of the buffer.

11460       If the *+line* option is specified, the current line indicator will be set to the specified position,  
 11461       where *line* can be a number (or "\$") or specified as */pattern* or *?pattern*. Preceding the pattern  
 11462       with a "/" will start a search from the beginning of the file. Preceding the pattern with a "?" will  
 11463       start a search from the end of the file. This command is affected by the editor options **autowrite**  
 11464       and **writeany**.

11465       **File**

11466       *Synopsis:*     f[ile] [file]

11467       Write the current pathname, the number of lines, and the current position when no *file* argument  
 11468       has been specified; *ex* may write other unspecified information. If no current file has yet been  
 11469       established, an unspecified message will be written to indicate that no file is being edited. With  
 11470       *file*, *ex* will change the current filename to *file* without changing the contents of the buffer or the  
 11471       previous current file.

11472       **Global**

11473       *Synopsis:*     [range] g[lobal] /pattern/ [commands]  
 11474       *Synopsis:*     [range] v /pattern/ [commands]

11475       Mark the lines within the given range that match (g) or do not match (v) the given pattern. Then  
 11476       execute the *ex* commands given by *commands* with the current line (.) set to each marked line.  
 11477       The *range* defaults to the entire file.

11478       Multiple *commands* can be specified, one per line, by escaping each newline character with a  
 11479       backslash. If *commands* are omitted, each line will be written. For the **append**, **change** and **insert**  
 11480       commands, the input text will be included as part of the **global** command line; in this case the  
 11481       terminating period can be omitted if it ends *commands*. The **visual** command can be specified as  
 11482       one of the *commands*. In this mode, input will be taken from the terminal. Entering a Q from  
 11483       visual mode causes the next line matching the pattern to be selected and visual mode to be  
 11484       reentered, until the list is exhausted.

11485       The **global** command itself and the **undo** command cannot be used in *commands*. The editor  
 11486       options **autoprint**, **autoindent** and **report** will be inhibited for the duration of the *g* or *v*  
 11487       command.

**Insert**

*Synopsis:*     [*line*] i[nsert][!]

Enter input mode; the input text will be placed before the specified line. The current line indicator will be set to the last line input; if no lines were input, it will be set to the line before the target line, or to the first line of the file if there are no lines preceding the target. Following the command name with the character "!" causes the **autoindent** editor option setting to be toggled for the duration of this command only.

**Join**

*Synopsis:*     [*range*] j[oin][!] [*count*] [*flags*]

Join the text from the specified lines together into one line. In the POSIX locale, when the last character on the first line of a pair of lines to be joined is a period, two space characters will be added following the period; when the last character of the first line is a blank character or when the first character on the second line of the pair is a ")", no space characters will be added; otherwise, one space character will be added following the last character of the first line. Extra blank characters at the start of a line will be discarded.

Appending a character "!" to the **join** command name causes a simpler join with no white-space processing, independent of the current locale.

**List**

*Synopsis:*     [*range*] l[ist] [*count*] [*flags*]

Write the addressed lines in a way that should be unambiguous: non-printable characters will be written as implementation-dependent multi-character sequences; the end of the line will be marked with a "\$".

Long lines will be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. The only useful flag is "#", for line numbers. The current line indicator will be set to the last line written.

**Map**

*Synopsis:*     map[!] [*x rhs*]

Define macros for use in visual mode. The first argument must be a single character or the sequence *#digit* (the latter meaning one of the terminal's numbered function keys). When this character or function key is typed in visual mode, the action will be as if the corresponding *rhs* had been typed. If the character "!" is appended to the command name **map**, the mapping is effective during input mode rather than command mode. This allows *x* to have two different macro definitions at the same time: one for command mode and one for input mode. Non-printable characters, except for a tab character, require escaping with a <control>-V (or <control>-Q) to be entered in the arguments. On certain block-mode terminals, the mapping need not occur immediately (for example, it may occur after the terminal transmits a group of characters to the system), but it will achieve the same results of modifying the file as if it occurred immediately. Implementations may restrict the set of commands accepted within *rhs*; the list of restrictions is implementation-dependent.

The **map** command with no arguments will write all of the macros currently defined. If "!" is appended to the command, only the macros effective during input mode will be written; otherwise, only the macros effective during command mode will be written.

11530 **Mark**11531 *Synopsis:* `[line] ma[rk] x`11532 *Synopsis:* `[line] k x`

11533 Give the specified line the specified mark *x*, which must be a single lower-case letter of the  
 11534 POSIX locale. The current line position will not be affected. The expression '*x*' can then be used  
 11535 as an address in any command requiring one. For example “*.,xd*” deletes all the lines from the  
 11536 current one to the marked line. Also see the *vi* “*”* and “*”* commands for uses of the mark in visual  
 11537 mode. If the '*x*' command is used in non-visual mode, the character marked will be the first  
 11538 non-blank character of the current line. Otherwise, the character marked will be the character at  
 11539 the current column of the current line.

11540 **Move**11541 *Synopsis:* `[range] m[ove] line`

11542 Move the specified lines (*range*) to be after the target line (*line*). The current line indicator will be  
 11543 set to the first of the moved lines.

11544 **Next**11545 *Synopsis:* `n[ext][!] [file ...]`

11546 Edit the next file from the argument list. If the current buffer has been modified since the last  
 11547 write, a warning will be written and the command will be aborted. This action can be  
 11548 overridden by appending the character “!” to the command name (*n!*). The argument list can be  
 11549 replaced by specifying a new one as operands to this command. Editing then starts with the first  
 11550 file on this new list. The current line indicator will be reset as described for the **edit** command.  
 11551 This command is affected by the editor options **autowrite** and **writeany**; see **Edit Options in ex**  
 11552 on page 325 for details.

11553 **Number**11554 *Synopsis:* `[range] nu[mber] [count] [flags]`11555 *Synopsis:* `[range] # [count] [flags]`

11556 Write the selected lines, each preceded with its line number in decimal. Non-printable  
 11557 characters, except for a tab character, will be expanded as specified by the **print** command. The  
 11558 format is as follows:

11559 `"%+6d\t%s", <line number>, <line text>`

11560 The only meaningful flag is **l**, which causes the additional expanded writing of tab characters  
 11561 and end-of-lines done by the **list** command to be performed. The current line indicator will be  
 11562 set to the last line written.

11563 **Open**11564 *Synopsis:* `[line] o[pen] /pattern/ [flags]`

11565 Enter open mode, which is equivalent to visual mode with a one-line window. All the visual  
 11566 mode commands are available. If a match is found for the optional regular expression in *line*, the  
 11567 cursor will be placed at the start of the matching pattern. The visual mode command **Q** (see *vi*)  
 11568 will exit open mode. This command need not be supported on block-mode terminals.

11569 **Preserve**11570 *Synopsis:*    pre[serve]

11571 Save the current buffer in a form that can later be recovered by using **ex -r** or by using the  
 11572 **recover** command. After the file has been preserved, a mail message will be sent to the user.  
 11573 This message can be read by invoking the *mailx* utility. The message will contain the name of  
 11574 the file, the time of preservation, and an **ex** command that could be used to recover the file.  
 11575 Additional information may be included in the mail message.

11576 **Print**11577 *Synopsis:*    [range] p[rint] [count] [flags]

11578 Write the addressed lines. Non-printable characters, except for the tab character, will be written  
 11579 as implementation-dependent multi-character sequences.

11580 Long lines will be folded; the length at which folding occurs is unspecified, but should be  
 11581 appropriate for the output device. The only meaningful *flags* are "#" and l. The current line  
 11582 indicator will be set to the last line written.

11583 **Put**11584 *Synopsis:*    [line] pu[t] [buffer]

11585 Put back deleted or yanked lines after line *line*. A buffer can be specified; otherwise, the text in  
 11586 the unnamed buffer (where deleted or yanked text is placed by default) will be restored. The  
 11587 current line indicator will be set to the first line put back.

11588 **Quit**11589 *Synopsis:*    q[uit][!]

11590 Terminate the editing session. If the current buffer has been modified since the last write, a  
 11591 warning will be written and the command will fail. This warning can be overridden and an exit  
 11592 forced, discarding changes, by appending the character "!" to the command name.

11593 **Read**11594 *Synopsis:*    [line] r[ead][!] [file]

11595 Place a copy of the specified file in the current buffer after the target line (which can be line 0 to  
 11596 place text at the beginning). If no *file* is named, the current file is the default. If there is no  
 11597 current file, then *file* will become the current file. If there is no current file nor *file* operand, the  
 11598 command will fail.

11599 The current line indicator will be set to the last line read. In visual mode, the current line  
 11600 indicator will be set to the first line read. If *file* is preceded by "!", *file* is taken to be an operating  
 11601 system command and passed to the program named in the *SHELL* environment variable; the  
 11602 resultant output will be read in to the buffer. The special meaning of "!" can be overridden by  
 11603 escaping it with a backslash character.

11604 **Recover**11605 *Synopsis:*    rec[over] file

11606 Attempt to recover *file* if it was saved as the result of a **preserve** command, the receipt of a signal  
 11607 (see the ASYNCHRONOUS EVENTS section), or a system or editor crash. The current line  
 11608 indicator will be reset as described for the **read** editor command.

11609 **Rewind**11610 *Synopsis:*    `rew[ind][!]`

11611 Rewind the argument list; that is, the current file will be set to the first file in the argument list.  
 11612 This is equivalent to a **next** command with the current argument list as its operands. If the  
 11613 current buffer has been modified since the last write, a warning will be written and the  
 11614 command will be aborted. The action can be overridden by appending the character "!" to the  
 11615 command name (rew!). The current line indicator will be reset as described for the **read** editor  
 11616 command. This command is affected by the editor options **autowrite** and **writeany**; see **Edit**  
 11617 **Options in ex** on page 325 for details.

11618 **Set**11619 *Synopsis:*    `se[t] [option[=[value]]...] [nooption...] [option?...] [all]`

11620 When no arguments are specified, write those options whose values have been changed from  
 11621 the default settings; when the argument **all** is specified, write all of the option values.

11622 Giving an option name followed by the character "?" causes the current value of that option to  
 11623 be written. The "?" can be separated from the option name by zero or more blank characters.  
 11624 The "?" is necessary only for Boolean valued options. Boolean options can be given values by  
 11625 the form **se option** to turn them on or **se nooption** to turn them off; string and numeric options  
 11626 can be assigned by the form **se option=value**. Blank characters in strings can be included as is by  
 11627 preceding each such character with a backslash. More than one option can be set or listed by a  
 11628 single **set** command by specifying multiple arguments, each separated from the next by one or  
 11629 more blank characters.

11630 See **Edit Options in ex** on page 325 for further details about options.

11631 **Shell**11632 *Synopsis:*    `sh[ell]`

11633 Invoke the program named in the *SHELL* environment variable with the argument **-i**  
 11634 (interactive mode). Editing will be resumed when the program exits.

11635 **Source**11636 *Synopsis:*    `so[urce] file`

11637 Read and execute commands from the file specified by the mandatory argument *file*. Such **so**  
 11638 commands can be nested. The maximum supported nesting depth is implementation-  
 11639 dependent, but will be at least one.

11640 **Substitute**11641 *Synopsis:*    `[range] s[ubstitute] [/pattern/repl/[options] [count] [flags]]`

11642 Replace the first instance of the pattern *pattern* by the string *repl* on each specified line. (See  
 11643 **Regular Expressions in ex** on page 324 and **Replacement Strings in ex** on page 324.) If the  
 11644 */pattern/repl/* argument is not present, the */pattern/repl/* from the previous **substitute** command  
 11645 will be used. If *options* includes the letter **g** (global), all non-overlapping instances of the pattern  
 11646 in the line will be substituted. If the option letter **c** (confirm) is included, then before each  
 11647 substitution the line will be written with ^ characters written on the following line, adjacent to  
 11648 and identifying the pattern to be replaced; an affirmative response causes the substitution to be  
 11649 done, while any other input aborts it. An affirmative response consists of a line with the  
 11650 affirmative response (as defined by the current locale) at the beginning of the line. Such a line

11651 will be subject to editing in the same way as the command line (the "/" or ":" line at the bottom  
 11652 of the screen). The current line indicator will be set to the last line substituted. When the **c**  
 11653 option is used, typing the interrupt character or receiving the SIGINT signal will stop the  
 11654 substitute operation and **ex** will return to command mode. All substitutions completed before  
 11655 the interrupt occurred will be retained, and none will be made after that point. The current line  
 11656 indicator will be set to the last line substituted. This command is affected by the *LC\_MESSAGES*  
 11657 environment variable and the **wraps** option.

## 11658 Suspend

11659 *Synopsis:*     su[suspend][!]

11660 *Synopsis:*     st[op][!]

11661 Allow control to return to the invoking process; **ex** will suspend itself as if it had received the  
 11662 SIGTSTP signal. The suspension will occur only if job control is enabled in the invoking shell  
 11663 (see the description of **set -m**).

11664 Following either **suspend** or **stop** with the character "!" will affect the operation of the **autowrite**  
 11665 editor option for this command only.

11666 The current **susp** character (see *stty*) will also cause the suspension.

## 11667 Tag

11668 *Synopsis:*     ta[g][!] tagstring

11669 Search for the *tagstring*, which can be in a different file. If the tag is in a different file, then the  
 11670 new file will be opened for editing. If the current buffer has been modified since the last write, a  
 11671 warning will be written and the command will be aborted. The action can be overridden by  
 11672 appending the character "!" to the command name. The current line indicator will be reset to the  
 11673 line indicated by the tag. This command is affected by the editor options **autowrite** and  
 11674 **writen**; see **Edit Options in ex** on page 325 and *ctags* for details. This command is affected by  
 11675 the **tags** editor option.

11676 The **tag** command will search for *tagstring* in the tag file referred to by the **tags** editor option  
 11677 until a reference to *tagstring* is found. The file pointed to by this reference will be loaded into the  
 11678 buffer, and the current line will be set to the first occurrence of the pattern specified in the tag  
 11679 file associated with the supplied *tagstring*; if the tag file contained a line-number reference, the  
 11680 current line will be set to that line. If the pattern or line number is not found, an error message  
 11681 will be written. If a file referred to by the **tags** editor option does not exist or is not readable, an  
 11682 error message will be written. The results are unspecified if the format of a tags file is not as  
 11683 specified by the *ctags* utility description.

## 11684 Unabbreviate

11685 *Synopsis:*     una[bbrev] word

11686 Delete *word* from the list of abbreviations, as described by the **abbrev** editor command.



11687 **Undo**11688 *Synopsis:*     u[ndo]

11689 Reverse the changes made by the previous editing command (one that changes the contents of  
 11690 the buffer). For this purpose, **global** and **visual** are considered single commands. An **undo** can  
 11691 itself be reversed. Commands that affect the external environment, such as **write**, **edit** and **next**,  
 11692 cannot be undone.

11693 **Unmap**11694 *Synopsis:*     unm[ap][!] x

11695 If no "!" is specified, remove the command-mode macro definition for x; otherwise, remove the  
 11696 input-mode macro definition for x. See the **map** command.

11697 **Visual**11698 *Synopsis:*     [line] vi[sual] [type] [count] [flags]

11699 Enter visual mode with the current line indicator set to *line*. The *type* is optional, and can be "-",  
 11700 ":", "+" or "^", as in the **z** command, to specify the position of the specified line on the screen  
 11701 window. (The default is to place the line at the top of the screen window.) A *count* specifies the  
 11702 number of lines that will initially be written; the default is the value of the editor option  
 11703 **window**. The command **Q** will exit visual mode. (For more information, see *vi*.) This command  
 11704 need not be supported on block-mode terminals.

11705 **Write**11706 *Synopsis:*     [range] w[rite][!] [>>] [file]11707 *Synopsis:*     [range] w[rite] [!] [file]11708 *Synopsis:*     [range] wq[!] [>>] [file]

11709 Write the specified lines (the whole buffer, if no *range* is given) out to the file represented by the  
 11710 pathname *file*, writing to standard output the number of lines and bytes written.

11711 If *file* is specified and is not the current file, and the file named by *file* exists, then the write will  
 11712 fail. If the current file has been changed by the **file** command and that file exists, the write will  
 11713 fail. In either case, the write can be forced by appending the character "!" to the command name.  
 11714 An existing file can be appended to by appending >> to the command name. If the file does not  
 11715 exist, the result is implementation-dependent.

11716 If the *file* is preceded by "!", the program named in the *SHELL* environment variable will be  
 11717 invoked with *file* as its second argument, and the specified lines will be passed as standard input  
 11718 to the command. The ! in this usage must be separated from the **w** command by at least one  
 11719 blank character. The special meaning of the ! can be overridden by escaping it with a backslash  
 11720 character. This command is affected by the editor options **writeln** and **readonly**.

11721 The command **wq** is equivalent to a **w** followed by a **q**; **wq!** is equivalent to **w!** followed by **q**.  
 11722 If the current buffer has no pathname associated with it, the **write** command will fail.

**Write and Exit**

*Synopsis:*     [*range*] x[*it*][!] [*file*]

Perform a **write** command if any changes have been made to the current buffer since the last write to any file. The *range* defaults to the entire file.

Unless the command fails because an attempt to write lines to a file did not succeed, the *ex* utility will exit after an **x** command. This command is affected by the editor options **writeln** and **readonly**.

**Yank**

*Synopsis:*     [*range*] ya[nk] [*buffer*] [*count*]

Place the specified lines in the named buffer. If no buffer is specified, the unnamed buffer will be used (where the most recently deleted or yanked text is placed by default).

**Adjust Window**

*Synopsis:*     [*line*] z [*type*] [*count*] [*flags*]

If *type* is omitted, then *count* lines following the specified line (*line*) will be written. The default for *count* is the value of the editor option **window**. The *type* argument will change the position at which *line* will be written on the screen by affecting the number of lines written before and after *line*.

If *type* is specified, it will be one of the following:

- Place *line* at the bottom of the screen.
- + Place *line* at the top of the screen.
- . Place *line* in the middle.
- ^ Write out *count* lines starting *count*\*2 lines before the addressed line; the net effect of this will be that a **z** command following another **z** command writes the previous page.
- = Centre the addressed line on the screen with a line of hyphens written immediately before and after it. The number of preceding and following lines of text written will be reduced to account for these lines of hyphens.

In all cases the current line indicator will be set to the last line written, with the exception of the "=" type, which causes the current line indicator to be set to that addressed in the command.

**Escape**

*Synopsis:*     ! *command*

*Synopsis:*     [*range*]! *command*

Pass the remainder of the line after the ! character to the program named in the *SHELL* environment variable for execution. A warning will be issued if the buffer has been changed since the last write. A single ! character will be written when the command completes. The current line position will not be affected.

Within the text of *command*, % and # will be expanded as pathnames (the current and alternative pathnames, respectively), and ! will be replaced with the text of the previous ! command. (Thus, !! will repeat the previous ! command.) If any such expansion is done, the expanded line will be echoed.

11762 The special meanings of %, # and ! can be overridden by escaping them with a backslash  
 11763 character. This command is affected by the editor options **autowrite** and **writeln**; see **Edit**  
 11764 **Options in ex** on page 325 for details.

11765 In the second form of the ! command, the remainder of the line after the ! will be passed to the  
 11766 program named in the *SHELL* environment variable, as described above. The specified lines will  
 11767 be provided to the program as standard input; the resulting output will replace the specified  
 11768 lines.

## 11769 **Shift Left**

11770 *Synopsis:*     [range] < [count] [flags]

11771 Shift the specified lines to the left; the number of character positions to be shifted will be  
 11772 determined by the editor option **shiftwidth**. Only leading blank characters will be lost in  
 11773 shifting; other characters will not be affected. The current line indicator will be set to the last line  
 11774 changed.

## 11775 **Shift Right**

11776 *Synopsis:*     [range] > [count] [flags]

11777 Shift the specified lines to the right, by inserting blank characters, using tab characters where  
 11778 possible, as determined by the editor option **shiftwidth**. Empty lines will not be changed. The  
 11779 current line indicator will be set to the last line changed.

## 11780 **Resubstitute**

11781 *Synopsis:*     [range] & [options] [count] [flags]

11782 *Synopsis:*     [range] s[ubstitute] [options] [count] [flags]

11783 *Synopsis:*     [range] ~ [options] [count] [flags]

11784 Repeat the previous substitute command, as if & were replaced by the previous:

11785         s/pattern/repl/

11786 command. The same effect can be obtained by omitting the:

11787         /pattern/repl/

11788 string in the **substitute** command. The version of the command using tilde will be the same as &  
 11789 and **s**, but the *pattern* used will be the last regular expression used in any command, not  
 11790 necessarily the one used in the last substitute command. For example, in the sequence:

11791         s/red/blue/

11792         /green

11793         ~

11794 the "" is equivalent to:

11795         s/green/blue/

11796 **Scroll**11797 *Synopsis:*    *eof*

11798 Write the next *n* lines, where *n* is the value of the editor option **scroll**. The command is invoked  
 11799 with the **eof** character (see the description of the *stty eof* character). The current line indicator  
 11800 will be set to the last line written. This command need not be supported on block-mode  
 11801 terminals.

11802 **Write Line Number**11803 *Synopsis:*    [*line*] = [*flags*]

11804 Write the line number of the specified line (default last line). The current line position will not  
 11805 be affected.

11806 **Execute**11807 *Synopsis:*    @ *buffer*11808 *Synopsis:*    \* *buffer*

11809 Execute each line of the named buffer as an *ex* command. If no buffer is specified or is specified  
 11810 as "@" or "\*", the last buffer executed will be used. If there is no last buffer, an error occurs.

11811 **Regular Expressions in ex**

11812 The *ex* utility supports the basic regular expressions described in the **XBD** specification, **Section**  
 11813 **7.3, Basic Regular Expressions**. A null RE (/) is equivalent to the last RE encountered.

11814 Regular expressions can be used in addresses to specify lines and, in some commands (for  
 11815 example, the **substitute** command), to specify portions of a line to be substituted.

11816 The following constructs can be used to enhance the basic regular expressions:

11817 \<    Match the beginning of a *word*. (See the definition of *word* at the beginning of  
 11818 **Command Descriptions in ex** on page 312.)

11819 \>    Match the end of a *word*.

11820 ~       Match the replacement part of the last **substitute** command. The tilde (~) character can  
 11821 be escaped in a regular expression to become a normal character with no special  
 11822 meaning.

11823 When the editor option **nomagic** is set, the only characters with special meanings are "^" at the  
 11824 beginning of a pattern, "\$" at the end of a pattern, and "\". The characters ".", "\*", "[", and "~" are  
 11825 treated as ordinary characters unless preceded by a "\"; when preceded by a "\" they regain their  
 11826 special meaning.

11827 **Replacement Strings in ex**

11828 The character & (\& if the editor option **nomagic** is set) in the replacement string will stand for  
 11829 the text matched by the pattern to be replaced. The character ~ (\~ if **nomagic** is set) will be  
 11830 replaced by the replacement part of the previous substitute command. The sequence \n, where  
 11831 *n* is an integer, will be replaced by the text matched by the pattern enclosed in the *n*th set of  
 11832 parentheses \ ( and \).

11833 The strings \l, \u, \L and \U can be used to modify the case of elements in the replacement  
 11834 string (using the \& or \<*digit*>) notation. The string \l (\u) causes the first character (actually  
 11835 inserted by the substitution) that follows the \l (\u) to be converted to lower-case (upper-case).  
 11836 The strings \L (\U) causes all characters subsequent to them to be converted to lower-case

11837 (upper-case) as they are inserted by the substitution until the string \e or \E, or the end of the  
 11838 replacement string, is encountered.

11839 An example of case conversion with the **s** command is as follows:

```
11840 :p
11841 The cat sat on the mat.
11842 :s/\<.at\>/\u&/gp
11843 The Cat Sat on the Mat.
11844 :s/S\(.*\)M/S\U\1\eM/p
11845 The Cat SAT ON THE Mat.
```

11846 In visual mode, a <control>-V <control>-M (or <control>-Q <control>-M) sequence in the  
 11847 replacement string will be mapped to a newline character, and so can be used to split lines. A  
 11848 literal <control>-M requires escaping by preceding it with a backslash (\^V^M or \^Q^M).

## 11849 Edit Options in ex

11850 The **ex** utility has a number of options that modify its behaviour. These options have default  
 11851 settings, which can be changed using the **set** command.

11852 Options are Boolean unless otherwise specified.

### 11853 **autoindent, ai**

11854 [Default *off*]

11855 If **autoindent** is set, each line in input mode will be indented (using first as many tab characters  
 11856 as possible, as determined by the editor option **tabstop**, and then using space characters) to align  
 11857 with the previous line. (Starting indentation will be determined by the line appended after, or  
 11858 the line inserted before or the first line changed, with an **a**, **i** or **c** command, respectively.) When  
 11859 a newline character is inserted in the middle of a line, and when **autoindent** is on, the first non-  
 11860 blank character to the right of the cursor will be aligned to the current margin on a new line  
 11861 immediately following the current line. Any blank characters to the left of the cursor position at  
 11862 which the newline character was entered will be retained. When **autoindent** is off, a new line  
 11863 will be created, but no blank characters will be discarded. Additional indentation can be  
 11864 provided as usual; succeeding lines will automatically be indented to the new alignment. A line  
 11865 entered during input mode with **autoindent** that contains no user-entered characters will be  
 11866 empty, despite the appearance of indentation during entry.

11867 Reducing the indent can be achieved when the cursor is at the current left margin by typing  
 11868 <control>-D one or more times; the cursor will be moved back to the previous integral number  
 11869 of **shiftwidth** spaces for each <control>-D. A ^ followed by a <control>-D will remove all  
 11870 indentation temporarily (for the current line); a 0 followed by a <control>-D will remove all  
 11871 indentation permanently (for the current line and subsequent lines until input mode is reentered  
 11872 or until the indentation is specifically set to some other value). Changing the indent with  
 11873 <control>-D need not be supported on block-mode terminals.

### 11874 **autoprint, ap**

11875 [Default *on*]

11876 If **autoprint** is set, the current line will be written after each command that changes buffer text.  
 11877 (Autoprint will be suppressed in the **global** [g and v] commands and for any command on  
 11878 which print operands [*flags*] are used to write explicitly the current line.)

|          |                                                                                                                                                            |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11879    | <b>autowrite, aw</b>                                                                                                                                       |
| 11880    | [Default <i>off</i> ]                                                                                                                                      |
| 11881    | If <b>autowrite</b> is set, when a <b>next</b> , <b>rewind</b> , <b>tag</b> , <b>edit</b> , <b>suspend</b> , <b>stop</b> or <b>!</b> command is given, the |
| 11882    | buffer will be written (to the current file) if it has been modified. Appending the character <b>!</b> to                                                  |
| 11883    | the command name for any of these commands except <b>!</b> causes the write not to occur. If the                                                           |
| 11884    | write fails, the command will be aborted and an error message will be written.                                                                             |
| 11885    | <b>beautify, bf</b>                                                                                                                                        |
| 11886    | [Default <i>off</i> ]                                                                                                                                      |
| 11887    | If <b>beautify</b> is set, all non-printable characters, other than tab, newline and form-feed characters,                                                 |
| 11888    | will be discarded from text read in from files.                                                                                                            |
| 11889    | <b>directory, dir</b>                                                                                                                                      |
| 11890 EX | [Default <i>implementation-dependent</i> ]                                                                                                                 |
| 11891    | The value of this option specifies the directory in which the editor buffer is to be placed. If this                                                       |
| 11892    | directory is not writable by the user, the editor quits.                                                                                                   |
| 11893    | <b>edcompatible, ed</b>                                                                                                                                    |
| 11894 EX | [Default <i>off</i> ]                                                                                                                                      |
| 11895    | Causes the presence of <b>g</b> and <b>c</b> suffixes on substitute commands to be remembered, and toggled                                                 |
| 11896    | by repeating the suffixes.                                                                                                                                 |
| 11897    | <b>errorbells, eb</b>                                                                                                                                      |
| 11898    | [Default <i>off</i> ]                                                                                                                                      |
| 11899    | If <b>errorbells</b> is set, error messages will be preceded by an alert action. Setting this option off                                                   |
| 11900    | causes the user to be informed of an error even when in visual mode, but rather than using the                                                             |
| 11901    | alert character, an error message will be written, using a standout mode of the terminal (such as                                                          |
| 11902    | inverse video) instead of the normal effect of the alert character, when the effect of the alert                                                           |
| 11903    | character is to cause the terminal to ring a bell or make other sounds. The editor should place                                                            |
| 11904    | the error message in a standout mode of the terminal, such as inverse video instead of ringing                                                             |
| 11905    | the bell, when the terminal capabilities allow this.                                                                                                       |
| 11906    | <b>exrc</b>                                                                                                                                                |
| 11907    | [Default <i>off</i> ]                                                                                                                                      |
| 11908    | If <b>exrc</b> is set, <b>ex</b> will access any <b>.exrc</b> file in the current directory, as described previously. If <b>exrc</b>                       |
| 11909    | is not set, <b>ex</b> will ignore any <b>.exrc</b> file in the current directory during initialisation, unless the                                         |
| 11910    | current directory is that named by the <i>HOME</i> variable.                                                                                               |
| 11911    | <b>ignorecase, ic</b>                                                                                                                                      |
| 11912    | [Default <i>off</i> ]                                                                                                                                      |
| 11913    | If <b>ignorecase</b> is set, characters that have upper-case and lower-case representations will have                                                      |
| 11914    | those representations considered as equivalent for purposes of regular expression comparison.                                                              |

|          |                                                                                                                         |
|----------|-------------------------------------------------------------------------------------------------------------------------|
| 11915    | <b>lisp</b>                                                                                                             |
| 11916 EX | [Default <i>off</i> ]                                                                                                   |
| 11917    | Autoindent mode and the (, ), {, }, [[ and ]] commands in visual mode are suitably modified for                         |
| 11918    | <b>lisp</b> code.                                                                                                       |
| 11919    | <b>list</b>                                                                                                             |
| 11920    | [Default <i>off</i> ]                                                                                                   |
| 11921    | If <b>list</b> is set, write the addressed lines in a way that should be unambiguous: non-printable                     |
| 11922    | characters will be written as implementation-dependent multi-character sequences; the end of                            |
| 11923    | the line will be marked with a "\$".                                                                                    |
| 11924    | <b>magic</b>                                                                                                            |
| 11925    | [Default <i>on</i> ]                                                                                                    |
| 11926    | If <b>magic</b> is set, change the interpretation of characters in regular expressions and substitution                 |
| 11927    | replacement strings (see <b>Regular Expressions in ex</b> on page 324 and <b>Replacement Strings in ex</b>              |
| 11928    | on page 324).                                                                                                           |
| 11929    | <b>mesg</b>                                                                                                             |
| 11930    | [Default <i>on</i> ]                                                                                                    |
| 11931    | If <b>mesg</b> is set, the permission for others to use the <b>write</b> or <b>talk</b> commands to write to the        |
| 11932    | terminal will be turned on while in visual mode. The shell-level command <i>mesg n</i> takes                            |
| 11933    | precedence over any setting of the <i>ex mesg</i> option; that is, if <b>mesg y</b> was issued before <i>ex</i> started |
| 11934    | (or in a shell escape), such as:                                                                                        |
| 11935    | : !mesg y                                                                                                               |
| 11936    | the <b>mesg</b> option in <i>ex</i> can suppress incoming messages, but the <b>mesg</b> option cannot enable            |
| 11937    | incoming messages if <b>mesg n</b> was issued.                                                                          |
| 11938    | <b>number, nu</b>                                                                                                       |
| 11939    | [Default <i>off</i> ]                                                                                                   |
| 11940    | If <b>number</b> is set, lines will be written with line numbers, as with the <b>number</b> command.                    |
| 11941    | <b>paragraphs, para</b>                                                                                                 |
| 11942    | [Default <i>implementation-dependent</i> ]                                                                              |
| 11943    | The <b>paragraph</b> option defines additional paragraph boundaries for the { and } commands in                         |
| 11944    | visual mode. The <b>paragraph</b> option can be set to a character string consisting of zero or more                    |
| 11945    | character pairs. The default value is implementation-dependent.                                                         |

11946 In the text to be edited, the character string `<newline>.<char-pair>`, where `<char-pair>` is one of  
 11947 the character pairs found in **paragraph**, defines a paragraph boundary. For example, if:

11948 `paragraph=LaA ##`

11949 then all of the following additional paragraph boundaries would be recognised:

11950 `<newline>.La`

11951 `<newline>.A<space>`

11952 `<newline>.##`

### 11953 **prompt**

11954 [Default *on*]

11955 If **prompt** is set, command mode input will be prompted for with a colon (:); when unset, no  
 11956 prompt will be written.

### 11957 **readonly**

11958 [Default *see text*]

11959 If **readonly** is set, read-only mode will be enabled. Writing to a different file will be allowed in  
 11960 read-only mode; in addition, the write can be forced by using the character "!" (see the editor  
 11961 command **write**). The default setting will be *off* unless the file lacks write permission or the  
 11962 command-line option **-R** is used.

### 11963 **redraw**

11964 EX [Default *off*]

11965 The editor simulates an intelligent terminal on a dumb terminal. (Since this is likely to require a  
 11966 large amount of output to the terminal, it is useful only at high transmission speeds.)

### 11967 **remap**

11968 [Default *on*]

11969 If **remap** is set, macro translation will allow for macros defined in terms of other macros;  
 11970 translation will continue until the final product is obtained. If unset, only a one-step translation  
 11971 will be done.

### 11972 **report**

11973 [Default 5]

11974 The value of this option will give the number of lines that can be changed by an editor command  
 11975 before a report is generated on the number of lines affected.

### 11976 **scroll**

11977 [Default `window/2`]

11978 The value of this option will determine the number of lines scrolled on a **eof** command (see  
 11979 **Scroll** on page 324 and the description of the *stty eof* character). Changing the value of **window**  
 11980 in *EXINIT*, one of the *.exrc* files, or with a **set** command will not affect the value of scroll. This  
 11981 editor option need not be supported on block-mode terminals.



|          |                                                                                                                                                                                                                                                                                                                      |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11982    | <b>sections</b>                                                                                                                                                                                                                                                                                                      |
| 11983    | [Default <i>implementation-dependent</i> ]                                                                                                                                                                                                                                                                           |
| 11984    | The <b>sections</b> option defines additional section boundaries for the [[ and ]] commands in visual mode. The <b>sections</b> option can be set to a character string consisting of zero or more character pairs. The default value is implementation-dependent.                                                   |
| 11985    |                                                                                                                                                                                                                                                                                                                      |
| 11986    |                                                                                                                                                                                                                                                                                                                      |
| 11987    | In the text to be edited, the character string <newline>.< <i>char-pair</i> >, where < <i>char-pair</i> > is one of the character pairs found in <b>sections</b> , defines a section boundary in the same manner that paragraph boundaries are defined. (See the <b>paragraphs</b> command.)                         |
| 11988    |                                                                                                                                                                                                                                                                                                                      |
| 11989    |                                                                                                                                                                                                                                                                                                                      |
| 11990    | <b>shell, sh</b>                                                                                                                                                                                                                                                                                                     |
| 11991    | [Default from the environment <i>SHELL</i> ]                                                                                                                                                                                                                                                                         |
| 11992    | The value of this option can be a string representing the pathname of the shell to be invoked for the "!" shell escape command, and by the <b>shell</b> command. The default is taken from the <i>SHELL</i> variable in the environment; see the ENVIRONMENT VARIABLES section for default values for <i>SHELL</i> . |
| 11993    |                                                                                                                                                                                                                                                                                                                      |
| 11994    |                                                                                                                                                                                                                                                                                                                      |
| 11995    |                                                                                                                                                                                                                                                                                                                      |
| 11996    | <b>shiftwidth, sw</b>                                                                                                                                                                                                                                                                                                |
| 11997    | [Default 8]                                                                                                                                                                                                                                                                                                          |
| 11998    | The value of this option gives the width of an indentation level used during <b>autoindent</b> and by the shift commands.                                                                                                                                                                                            |
| 11999    |                                                                                                                                                                                                                                                                                                                      |
| 12000    | <b>showmatch, sm</b>                                                                                                                                                                                                                                                                                                 |
| 12001    | [Default <i>off</i> ]                                                                                                                                                                                                                                                                                                |
| 12002    | If <b>showmatch</b> is set, in visual mode, when a ")" or "]" is typed, the matching "(" or "{" will be shown if it is still on the screen. This editor option need not be supported on block-mode terminals.                                                                                                        |
| 12003    |                                                                                                                                                                                                                                                                                                                      |
| 12004    |                                                                                                                                                                                                                                                                                                                      |
| 12005    | <b>showmode</b>                                                                                                                                                                                                                                                                                                      |
| 12006    | [Default <i>off</i> ]                                                                                                                                                                                                                                                                                                |
| 12007    | If <b>showmode</b> is set, in visual mode, the current mode that the editor is in will be written on the last line of the screen. Modes that will be reported are command mode and input mode; other unspecified modes may be written.                                                                               |
| 12008    |                                                                                                                                                                                                                                                                                                                      |
| 12009    |                                                                                                                                                                                                                                                                                                                      |
| 12010    | <b>slowopen</b>                                                                                                                                                                                                                                                                                                      |
| 12011 EX | [Default <i>off</i> ]                                                                                                                                                                                                                                                                                                |
| 12012    | In visual mode, this option prevents screen updates during input to improve throughput on unintelligent terminals.                                                                                                                                                                                                   |
| 12013    |                                                                                                                                                                                                                                                                                                                      |

|       |                                                                                                                     |
|-------|---------------------------------------------------------------------------------------------------------------------|
| 12014 | <b>tabstop, ts</b>                                                                                                  |
| 12015 | [Default 8]                                                                                                         |
| 12016 | The value of this option specifies the software tab stops to be used by the editor to expand tabs                   |
| 12017 | in the input.                                                                                                       |
| 12018 | <b>tags</b>                                                                                                         |
| 12019 | [Default <i>see text</i> ]                                                                                          |
| 12020 | The value of this option can be a string representing space-character-separated pathnames that                      |
| 12021 | will be used as tag files for the <b>tag</b> command. A requested tag will be searched for sequentially             |
| 12022 | in the specified files. By default, filenames of <b>tags</b> will be searched for in the current directory          |
| 12023 | and in other implementation-dependent directories.                                                                  |
| 12024 | <b>term</b>                                                                                                         |
| 12025 | [Default from the environment <i>TERM</i> ]                                                                         |
| 12026 | The value of this option can be a string representing the terminal type of the output device. The                   |
| 12027 | default is taken from the <i>TERM</i> variable in the environment; see the ENVIRONMENT                              |
| 12028 | VARIABLES section for default values for <i>TERM</i> .                                                              |
| 12029 | <b>terse</b>                                                                                                        |
| 12030 | [Default <i>off</i> ]                                                                                               |
| 12031 | If <b>terse</b> is set, error messages may be less verbose. However, except for this caveat, error                  |
| 12032 | messages are unspecified. Furthermore, not all error messages need change for different settings                    |
| 12033 | of this option.                                                                                                     |
| 12034 | <b>warn</b>                                                                                                         |
| 12035 | [Default <i>on</i> ]                                                                                                |
| 12036 | If <b>warn</b> is set, <i>ex</i> will write a warning message to standard error if the contents of the buffer have  |
| 12037 | not been saved before a "!" command escape.                                                                         |
| 12038 | <b>window</b>                                                                                                       |
| 12039 | [Default <i>see text</i> ]                                                                                          |
| 12040 | The value of this option determines the default number of lines in a screenful, as written by the <b>z</b>          |
| 12041 | command. When in visual mode, the number of lines output when moving up or down the file                            |
| 12042 | by a <i>screenful</i> . The value of <b>window</b> can be unrelated to the real screen size, except that it will be |
| 12043 | set on entry to be the current number of screen lines. (The current number of screen lines will be                  |
| 12044 | determined by the system or overridden by the user, as described for <i>LINES</i> in the                            |
| 12045 | ENVIRONMENT VARIABLES section and the <b>XBD</b> specification, <b>Chapter 6, Environment</b>                       |
| 12046 | <b>Variables</b> .) The baud rate of the terminal line may reduce the default in an implementation-                 |
| 12047 | dependent manner. The default value of <b>windows</b> also can be overridden by specifying a                        |
| 12048 | window size using the <b>-w</b> command-line option.                                                                |

12049       **wrapscan, ws**

12050       [Default *on*]

12051       If **wrapscan** is set, searches (using *//* or *??*) will wrap around the end of the editing buffer; when  
 12052       unset, searches will stop at the beginning of the editing buffer for *??*, or at the end of the editing  
 12053       buffer for *//*.

12054       **wrapmargin, wm**

12055       [Default 0]

12056       If the value of this option is greater than zero (say *n*) in visual mode during text entry, then, in  
 12057       the POSIX locale, a newline character will replace all consecutive blank characters, at the  
 12058       boundary between a blank character and a non-blank character, so that lines will end at least *n*  
 12059       spaces from the ending margin of the terminal screen. (The ending margin will be determined  
 12060       by the system or overridden by the user, as described for *COLUMNS* in the ENVIRONMENT  
 12061       VARIABLES section and the **XBD** specification, **Chapter 6, Environment Variables**.) If a line  
 12062       consists of a sequence of non-blank characters long enough such that it extends continuously  
 12063       from the beginning margin to beyond the ending margin, that sequence will not be broken by the  
 12064       action of this option. If the value is zero, no wrapping will be performed.

12065       **writeany, wa**

12066       [Default *off*]

12067       If **writeany** is set, file-overwriting checks will be inhibited that would otherwise be made before  
 12068       **write** and **xit** commands, or before an automatic write (see editor option **autowrite**), allowing a  
 12069       write to any file (provided permissions allow it).

## 12070 EXIT STATUS

12071       The following exit values are returned:

- 12072       0   Successful completion.
- 12073       >0   An error occurred.

## 12074 CONSEQUENCES OF ERRORS

12075       When an error in the input script is encountered, or when an error is detected that is a  
 12076       consequence of the data (not) present in the file or due to an external condition such as a read or  
 12077       write error:

- 12078       • If the standard input is a terminal device file, all input will be flushed, and a new command  
 12079       read.
- 12080       • If the standard input is a regular file, **ex** will terminate with a non-zero exit status.
- 12081       • If the command in error was one of those specified with the command-line **-c** option (or the  
 12082       obsolescent **+** option), all of the remaining **-c** commands will be flushed, and a new  
 12083       command read from standard input.

## 12084 APPLICATION USAGE

12085       If a SIGSEGV signal is received while **ex** is saving a file, the file might not be successfully saved.

12086       The **next** command can accept more than one file, so usage such as:

12087       next `ls [abc]\*`

12088       is valid; it would not be valid for the **edit** or **read** commands, for example, because they expect  
 12089       only one file and unspecified results occur.

12090 **EXAMPLES**

12091 None.

12092 **FUTURE DIRECTIONS**

12093 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
12094 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
12095 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
12096 finalised.

12097 **SEE ALSO**12098 *ed, sed, vi.*12099 **CHANGE HISTORY**

12100 First released in Issue 2.

12101 **Issue 4**

12102 Aligned with the ISO/IEC 9945-2: 1993 standard.

12103 **Issue 5**

12104 FUTURE DIRECTIONS section added.

12105 **NAME**

12106           expand — convert tabs to spaces

12107 **SYNOPSIS**12108           expand [-t *tablist*][*file* ...]12109 OB       expand [-*tabstop*][-*tab1*,*tab2*,...,*tabn*][*file*...]12110 **DESCRIPTION**

12111       The *expand* utility writes files or the standard input to the standard output with tab characters  
 12112       replaced with one or more space characters needed to pad to the next tab stop. Any backspace  
 12113       characters will be copied to the output and cause the column position count for tab stop  
 12114       calculations to be decremented; the column position count will not be decremented below zero.

12115 **OPTIONS**

12116 OB       The *expand* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**; the  
 12117       obsolescent version does not.

12118       The following options are supported:

12119       -t *tablist*

12120           Specify the tab stops. The argument *tablist* must consist of a single positive decimal  
 12121           integer or multiple positive decimal integers, separated by blank characters or commas,  
 12122           in ascending order. If a single number is given, tabs will be set *tablist* column positions  
 12123           apart instead of the default 8. If multiple numbers are given, the tabs will be set at  
 12124           those specific column positions.

12125           Each tab-stop position *N* must be an integer value greater than zero, and the list must  
 12126           be in strictly ascending order. This is taken to mean that, from the start of a line of  
 12127           output, tabbing to position *N* causes the next character output to be in the (*N*+1)th  
 12128           column position on that line.

12129           In the event of *expand* having to process a tab character at a position beyond the last of  
 12130           those specified in a multiple tab-stop list, the tab character is replaced by a single space  
 12131           character in the output.

12132 OB       In the obsolescent version, the single number is specified as *tabstop* with a leading minus;  
 12133       multiple tab stops are specified after a leading minus as *tab1*, *tab2* and so on.

12134 **OPERANDS**

12135       The following operand is supported:

12136       *file*       The pathname of a text file to be used as input.12137 **STDIN**

12138       See the INPUT FILES section.

12139 **INPUT FILES**

12140       Input files must be text files.

12141 **ENVIRONMENT VARIABLES**12142       The following environment variables affect the execution of *expand*:

12143       *LANG*       Provide a default value for the internationalisation variables that are unset or null. If  
 12144       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 12145       default locale will be used. If any of the internationalisation variables contains an  
 12146       invalid setting, the utility will behave as if none of the variables had been defined.

12147 **LC\_ALL**  
 12148 If set to a non-empty string value, override the values of all the other  
 12149 internationalisation variables.

12150 **LC\_CTYPE**  
 12151 Determine the locale for the interpretation of sequences of bytes of text data as  
 12152 characters (for example, single- as opposed to multi-byte characters in arguments and  
 12153 input files), the processing of tab and space characters, and for the determination of the  
 12154 width in column positions each character would occupy on a constant-width font  
 12155 output device.

12156 **LC\_MESSAGES**  
 12157 Determine the locale that should be used to affect the format and contents of diagnostic  
 12158 messages written to standard error.

12159 EX **NLSPATH**  
 12160 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

12161 **ASYNCHRONOUS EVENTS**  
 12162 Default.

12163 **STDOUT**  
 12164 The standard output is equivalent to the input files with tab characters converted into the  
 12165 appropriate number of space characters.

12166 **STDERR**  
 12167 Used only for diagnostic messages.

12168 **OUTPUT FILES**  
 12169 None.

12170 **EXTENDED DESCRIPTION**  
 12171 None.

12172 **EXIT STATUS**  
 12173 The following exit values are returned:  
 12174 0 Successful completion  
 12175 >0 An error occurred.

12176 **CONSEQUENCES OF ERRORS**  
 12177 The *expand* utility will terminate with an error message and non-zero exit status upon  
 12178 encountering difficulties accessing one of the *file* operands.

12179 **APPLICATION USAGE**  
 12180 None.

12181 **EXAMPLES**  
 12182 None.

12183 **FUTURE DIRECTIONS**  
 12184 None.

12185 **SEE ALSO**  
 12186 *tabs*, *unexpand*.

12187 **CHANGE HISTORY**  
 12188 First released in Issue 4.

12189 **NAME**12190        *expr* — evaluate arguments as an expression12191 **SYNOPSIS**12192        *expr operand*12193 **DESCRIPTION**12194        The *expr* utility will evaluate an expression and write the result to standard output.12195 **OPTIONS**

12196        None.

12197 **OPERANDS**12198        The single expression evaluated by *expr* will be formed from the operands, as described in the  
12199        EXTENDED DESCRIPTION section. Each of the expression operator symbols:

12200        (    )    |    &amp;    =    &gt;    &gt;=    &lt;    &lt;=    !=    +    -    \*    /    %    :

12201        and the symbols *integer* and *string* in the table must be provided as separate arguments to *expr*.12202 **STDIN**

12203        Not used.

12204 **INPUT FILES**

12205        None.

12206 **ENVIRONMENT VARIABLES**12207        The following environment variables affect the execution of *expr*:12208        **LANG**    Provide a default value for the internationalisation variables that are unset or null. If  
12209        *LANG* is unset or null, the corresponding value from the implementation-dependent  
12210        default locale will be used. If any of the internationalisation variables contains an  
12211        invalid setting, the utility will behave as if none of the variables had been defined.12212        **LC\_ALL**12213        If set to a non-empty string value, override the values of all the other  
12214        internationalisation variables.12215        **LC\_COLLATE**12216        Determine the locale for the behaviour of ranges, equivalence classes and multi-  
12217        character collating elements within regular expressions and by the string comparison  
12218        operators.12219        **LC\_CTYPE**12220        Determine the locale for the interpretation of sequences of bytes of text data as  
12221        characters (for example, single- as opposed to multi-byte characters in arguments) and  
12222        the behaviour of character classes within regular expressions.12223        **LC\_MESSAGES**12224        Determine the locale that should be used to affect the format and contents of diagnostic  
12225        messages written to standard error.12226 EX        **NLSPATH**12227        Determine the location of message catalogues for the processing of *LC\_MESSAGES*.12228 **ASYNCHRONOUS EVENTS**

12229        Default.

12230 **STDOUT**

12231 The *expr* utility will evaluate the expression and write the result to standard output. The  
 12232 character 0 will be written to indicate a zero value and nothing will be written to indicate a null  
 12233 string.

12234 **STDERR**

12235 Used only for diagnostic messages.

12236 **OUTPUT FILES**

12237 None.

12238 **EXTENDED DESCRIPTION**

12239 The formation of the expression to be evaluated is shown in the following table. The symbols  
 12240 *expr*, *expr1* and *expr2* represent expressions formed from *integer* and *string* symbols and the  
 12241 expression operator symbols (all separate arguments) by recursive application of the constructs  
 12242 described in the table. The expressions are listed in order of increasing precedence, with equal-  
 12243 precedence operators grouped between horizontal lines. All of the operators are left-associative.  
 12244

| Expression                                                                                                                                                                                | Description                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>expr1</i>   <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if it is neither null nor zero; otherwise, returns the evaluation of <i>expr2</i> .                                                                                                                                                                                                                                                                                      |
| <i>expr1</i> & <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if neither expression evaluates to null or zero; otherwise, returns zero.                                                                                                                                                                                                                                                                                                |
| <i>expr1</i> = <i>expr2</i><br><i>expr1</i> > <i>expr2</i><br><i>expr1</i> >= <i>expr2</i><br><i>expr1</i> < <i>expr2</i><br><i>expr1</i> <= <i>expr2</i><br><i>expr1</i> != <i>expr2</i> | Returns the result of a decimal integer comparison if both arguments are integers; otherwise, returns the result of a string comparison using the locale-specific collation sequence. The result of each comparison will be 1 if the specified relationship is true, or 0 if the relationship is false.<br>Equal.<br>Greater than.<br>Greater than or equal.<br>Less than.<br>Less than or equal.<br>Not equal. |
| <i>expr1</i> + <i>expr2</i><br><i>expr1</i> - <i>expr2</i>                                                                                                                                | Addition of decimal integer-valued arguments.<br>Subtraction of decimal integer-valued arguments.                                                                                                                                                                                                                                                                                                               |
| <i>expr1</i> * <i>expr2</i><br><i>expr1</i> / <i>expr2</i><br><i>expr1</i> % <i>expr2</i>                                                                                                 | Multiplication of decimal integer-valued arguments.<br>Integer division of decimal integer-valued arguments, producing an integer result.<br>Remainder of integer division of decimal integer-valued arguments.                                                                                                                                                                                                 |
| <i>expr1</i> : <i>expr2</i>                                                                                                                                                               | Matching expression. See below.                                                                                                                                                                                                                                                                                                                                                                                 |
| ( <i>expr</i> )                                                                                                                                                                           | Grouping symbols. Any expression can be placed within parentheses. Parentheses can be nested to a depth of {EXPR_NEST_MAX}.                                                                                                                                                                                                                                                                                     |
| <i>integer</i><br><i>string</i>                                                                                                                                                           | An argument consisting only of an (optional) unary minus followed by digits.<br>A string argument. See below.                                                                                                                                                                                                                                                                                                   |



12274 **Matching Expression**

12275 The ":" matching operator will compare the string resulting from the evaluation of *expr1* with the  
 12276 regular expression pattern resulting from the evaluation of *expr2*. Regular expression syntax is  
 12277 that defined in the XBD specification, **Section 7.3, Basic Regular Expressions**, except that all  
 12278 patterns are anchored to the beginning of the string (that is, only sequences starting at the first  
 12279 character of a string will be matched by the regular expression) and, therefore, it is unspecified  
 12280 whether "^" is a special character in that context. Usually, the matching operator will return a  
 12281 string representing the number of characters matched ("0" on failure). Alternatively, if the  
 12282 pattern contains at least one regular expression subexpression `[...(\\)]`, the string  
 12283 corresponding to `\\1` will be returned.

12284 **String Operand**

12285 A string argument is an argument that cannot be identified as an *integer* argument or as one of  
 12286 the expression operator symbols shown in the OPERANDS section.

12287 The use of string arguments **length**, **substr**, **index** or **match** produces unspecified results.

12288 **EXIT STATUS**

12289 The following exit values are returned:

- 12290 0 The *expression* evaluates to neither null nor zero.
- 12291 1 The *expression* evaluates to null or zero.
- 12292 2 Invalid *expression*.
- 12293 >2 An error occurred.

12294 **CONSEQUENCES OF ERRORS**

12295 Default.

12296 **APPLICATION USAGE**

12297 After argument processing by the shell, *expr* is not required to be able to tell the difference  
 12298 between an operator and an operand except by the value. If **\$a** is "=", the command:

12299 `expr $a = '='`

12300 looks like:

12301 `expr = = =`

12302 as the arguments are passed to *expr* (and they all may be taken as the "=" operator). The  
 12303 following works reliably:

12304 `expr X$a = X=`

12305 Also note that this specification permits implementations to extend utilities. The *expr* utility  
 12306 permits the integer arguments to be preceded with a unary minus. This means that an integer  
 12307 argument could look like an option. Therefore, the portable application must employ the --  
 12308 construct of Guideline 10 of the XBD specification, **Section 10.2, Utility Syntax Guidelines** to  
 12309 protect its operands if there is any chance the first operand might be a negative integer (or any  
 12310 string with a leading minus).

12311 **EXAMPLES**

12312 The *expr* utility has a rather difficult syntax:

- 12313 • Many of the operators are also shell control operators or reserved words, so they have to be  
 12314 escaped on the command line.

- Each part of the expression is composed of separate arguments, so liberal usage of blank characters is required. For example:

| Invalid            | Valid                 |
|--------------------|-----------------------|
| expr 1+2           | expr 1 + 2            |
| expr "1 + 2"       | expr 1 + 2            |
| expr 1 + ( 2 * 3 ) | expr 1 + \( 2 \* 3 \) |

In many cases, the arithmetic and string features provided as part of the shell command language are easier to use than their equivalents in *expr*. Newly written scripts should avoid *expr* in favour of the new features within the shell. See Section 2.5 on page 27 and Section 2.6.4 on page 37.

The following command:

```
a=$(expr $a + 1)
```

adds 1 to the variable *a*.

The following command, for *\$a* equal to either */usr/abc/file* or just *file*:

```
expr $a : '.*\/(.*\)' \| $a
```

returns the last segment of a pathname (that is, *file*). Applications should avoid the character *"/* used alone as an argument: *expr* may interpret it as the division operator.

The following command:

```
expr "//$a" : '.*\/(.*\)'
```

is a better representation of the previous example. The addition of the *//* characters eliminates any ambiguity about the division operator and simplifies the whole expression. Also note that pathnames may contain characters contained in the *IFS* variable and should be quoted to avoid having *\$a* expand into multiple arguments.

The following command:

```
expr "$VAR" : '.*'
```

returns the number of characters in *VAR*.

## FUTURE DIRECTIONS

The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the corrections. A future revision of this specification will align with IEEE Std. 1003.2b when finalised.

## SEE ALSO

Section 2.6.4 on page 37.

## CHANGE HISTORY

First released in Issue 2.

### Issue 4

Aligned with the ISO/IEC 9945-2: 1993 standard.

### Issue 5

FUTURE DIRECTIONS section added.

12355 **NAME**

12356 false — return false value

12357 **SYNOPSIS**

12358 false

12359 **DESCRIPTION**12360 The *false* utility will return with a non-zero exit code.12361 **OPTIONS**

12362 None.

12363 **OPERANDS**

12364 None.

12365 **STDIN**

12366 Not used.

12367 **INPUT FILES**

12368 None.

12369 **ENVIRONMENT VARIABLES**

12370 None.

12371 **ASYNCHRONOUS EVENTS**

12372 Default.

12373 **STDOUT**

12374 Not used.

12375 **STDERR**

12376 None.

12377 **OUTPUT FILES**

12378 None.

12379 **EXTENDED DESCRIPTION**

12380 None.

12381 **EXIT STATUS**12382 The *false* utility always will exit with a value other than zero.12383 **CONSEQUENCES OF ERRORS**

12384 Default.

12385 **APPLICATION USAGE**

12386 None.

12387 **EXAMPLES**

12388 None.

12389 **FUTURE DIRECTIONS**

12390 None.

12391 **SEE ALSO**12392 *true*.

12393 **CHANGE HISTORY**

12394 First released in Issue 2.

12395 **Issue 4**

12396 Aligned with the ISO/IEC 9945-2: 1993 standard.

12397 **NAME**12398        **fc** — process the command history list12399 **SYNOPSIS**12400        **fc** [-r][-e *editor*] [*first*[*last*]]12401        **fc** -l[-nr] [*first*[*last*]]12402        **fc** -s[*old=new*][*first*]12403 **DESCRIPTION**12404        The **fc** utility lists or edits and reexecutes, commands previously entered to an interactive *sh*.

12405        The command history list references commands by number. The first number in the list is  
 12406        selected arbitrarily. The relationship of a number to its command will not change except when  
 12407        the user logs in and no other process is accessing the list, at which time the system may reset the  
 12408        numbering to start the oldest retained command at another number (usually 1). When the  
 12409        number reaches an implementation-dependent upper limit, which will be no smaller than the  
 12410        value in *HISTSIZE* or 32 767 (whichever is greater), the shell may wrap the numbers, starting the  
 12411        next command with a lower number (usually 1). However, despite this optional wrapping of  
 12412        numbers, **fc** will maintain the time-ordering sequence of the commands. For example, if four  
 12413        commands in sequence are given the numbers 32 766, 32 767, 1 (wrapped), and 2 as they are  
 12414        executed, command 32 767 is considered the command previous to 1, even though its number is  
 12415        higher.

12416        When commands are edited (when the **-l** option is not specified), the resulting lines will be  
 12417        entered at the end of the history list and then reexecuted by *sh*. The **fc** command that caused the  
 12418        editing will not be entered into the history list. If the editor returns a non-zero exit status, this  
 12419        will suppress the entry into the history list and the command reexecution. Any command-line  
 12420        variable assignments or redirection operators used with **fc** will affect both the **fc** command itself  
 12421        as well as the command that results, for example:

12422        **fc** -s -- -l 2>/dev/null

12423        reinvokes the previous command, suppressing standard error for both **fc** and the previous  
 12424        command.

12425 **OPTIONS**12426        The **fc** utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

12427        The following options are supported:

12428        **-e** *editor*

12429            Use the editor named by *editor* to edit the commands. The *editor* string is a utility name,  
 12430            subject to search via the *PATH* variable (see the **XBD** specification, **Chapter 6,**  
 12431            **Environment Variables**). The value in the *FCEDIT* variable is used as a default when  
 12432            **-e** is not specified. If *FCEDIT* is null or unset, *ed* will be used as the editor.

12433        **-l**        (The letter ell.) List the commands rather than invoking an editor on them. The  
 12434        commands will be written in the sequence indicated by the *first* and *last* operands, as  
 12435        affected by **-r**, with each command preceded by the command number.

12436        **-n**        Suppress command numbers when listing with **-l**.12437        **-r**        Reverse the order of the commands listed (with **-l**) or edited (with neither **-l** nor **-s**).

12438            **-s**            Reexecute the command without invoking an editor.

# 12439 OPERANDS

12440            The following operands are supported:

12441            *first*

12442            *last*            Select the commands to list or edit. The number of previous commands that can be  
12443            accessed is determined by the value of the *HISTSIZE* variable. The value of *first* or *last*  
12444            or both will be one of the following:

12445            **[+]** *number*

12446                            A positive number representing a command number; command numbers can  
12447                            be displayed with the **-l** option.

12448            **-***number*

12449                            A negative decimal number representing the command that was executed  
12450                            *number* of commands previously. For example, **-1** is the immediately previous  
12451                            command.

12452            *string*            A string indicating the most recently entered command that begins with that  
12453            string. If the *old=new* operand is not also specified with **-s**, the string form of  
12454            the *first* operand cannot contain an embedded equal sign.

12455            When the synopsis form with **-s** is used:

- 12456                            • If *first* is omitted, the previous command will be used.

12457            For the synopsis forms without **-s**:

- 12458                            • If *last* is omitted, *last* defaults to the previous command when **-l** is specified;  
12459                            otherwise, it defaults to *first*.
- 12460                            • If *first* and *last* are both omitted, the previous 16 commands will be listed or the  
12461                            previous single command will be edited (based on the **-l** option).
- 12462                            • If *first* and *last* are both present, all of the commands from *first* to *last* will be edited  
12463                            (without **-l**) or listed (with **-l**). Editing multiple commands will be accomplished  
12464                            by presenting to the editor all of the commands at one time, each command starting  
12465                            on a new line. If *first* represents a newer command than *last*, the commands will be  
12466                            listed or edited in reverse sequence, equivalent to using **-r**. For example, the  
12467                            following commands on the first line are equivalent to the corresponding  
12468                            commands on the second:

12469                            `fc -r 10 20      fc      30 40`  
12470                            `fc      20 10      fc -r 40 30`

- 12471                            • When a range of commands is used, it will not be an error to specify *first* or *last*  
12472                            values that are not in the history list; *fc* will substitute the value representing the  
12473                            oldest or newest command in the list, as appropriate. For example, if there are only  
12474                            ten commands in the history list, numbered 1 to 10:

12475                            `fc -l`  
12476                            `fc 1 99`

12477                            will list and edit, respectively, all ten commands.

12478            **old=new**

12479                            Replace the first occurrence of string *old* in the commands to be reexecuted by the  
12480                            string *new*.

12481 **STDIN**

12482 Not used.

12483 **INPUT FILES**

12484 None.

12485 **ENVIRONMENT VARIABLES**12486 The following environment variables affect the execution of *fc*:12487 ***FCEDIT***

12488 This variable, when expanded by the shell, determines the default value for the  
 12489 **-e** *editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* will be used as  
 12490 the editor.

12491 ***HISTFILE***

12492 Determine a pathname naming a command history file. If the *HISTFILE* variable is not  
 12493 set, the shell may attempt to access or create a file **.sh\_history** in the user's home  
 12494 directory. If the shell cannot obtain both read and write access to, or create, the history  
 12495 file, it will use an unspecified mechanism that allows the history to operate properly.  
 12496 (References to history "file" in this section are understood to mean this unspecified  
 12497 mechanism in such cases.) An implementation may choose to access this variable only  
 12498 when initialising the history file; this initialisation will occur when *fc* or *sh* first attempt  
 12499 to retrieve entries from, or add entries to, the file, as the result of commands issued by  
 12500 the user, the file named by the *ENV* variable, or implementation-dependent system  
 12501 startup files. (The initialisation process for the history file can be dependent on the  
 12502 system startup files, in that they may contain commands that will effectively preempt  
 12503 the user's settings of *HISTFILE* and *HISTSIZE*. For example, function definition  
 12504 commands are recorded in the history file, unless the **set -o nolog** option is set. If the  
 12505 system administrator includes function definitions in some system startup file called  
 12506 before the *ENV* file, the history file will be initialised before the user gets a chance to  
 12507 influence its characteristics.) In some historical shells, the history file is initialised just  
 12508 after the *ENV* file has been processed. Therefore, it is implementation-dependent  
 12509 whether changes made to *HISTFILE* after the history file has been initialised are  
 12510 effective. Implementations may choose to disable the history list mechanism for users  
 12511 with appropriate privileges who do not set *HISTFILE*; the specific circumstances under  
 12512 which this will occur are implementation-dependent. If more than one instance of the  
 12513 shell is using the same history file, it is unspecified how updates to the history file from  
 12514 those shells interact. As entries are deleted from the history file, they will be deleted  
 12515 oldest first. It is unspecified when history file entries are physically removed from the  
 12516 history file.

12517 ***HISTSIZE***

12518 Determine a decimal number representing the limit to the number of previous  
 12519 commands that are accessible. If this variable is unset, an unspecified default greater  
 12520 than or equal to 128 will be used. The maximum number of commands in the history  
 12521 list is unspecified, but will be at least 128. An implementation may choose to access  
 12522 this variable only when initialising the history file, as described under *HISTFILE*.  
 12523 Therefore, it is unspecified whether changes made to *HISTSIZE* after the history file has  
 12524 been initialised are effective.

12525 ***LANG*** Provide a default value for the internationalisation variables that are unset or null. If  
 12526 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 12527 default locale will be used. If any of the internationalisation variables contains an  
 12528 invalid setting, the utility will behave as if none of the variables had been defined.

12529 **LC\_ALL**  
 12530 If set to a non-empty string value, override the values of all the other  
 12531 internationalisation variables.

12532 **LC\_CTYPE**  
 12533 Determine the locale for the interpretation of sequences of bytes of text data as  
 12534 characters (for example, single- as opposed to multi-byte characters in arguments and  
 12535 input files).

12536 **LC\_MESSAGES**  
 12537 Determine the locale that should be used to affect the format and contents of diagnostic  
 12538 messages written to standard error.

12539 EX **NLSPATH**  
 12540 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

12541 **ASYNCHRONOUS EVENTS**  
 12542 Default.

12543 **STDOUT**  
 12544 When the **-l** option is used to list commands, the format of each command in the list is as  
 12545 follows:  
 12546 `"%d\t%s\n", <line number>, <command>`  
 12547 If both the **-l** and **-n** options are specified, the format of each command is:  
 12548 `"\t%s\n", <command>`  
 12549 If the *<command>* consists of more than one line, the lines after the first are displayed as:  
 12550 `"\t%s\n", <continued-command>`

12551 **STDERR**  
 12552 Used only for diagnostic messages.

12553 **OUTPUT FILES**  
 12554 None.

12555 **EXTENDED DESCRIPTION**  
 12556 None.

12557 **EXIT STATUS**  
 12558 The following exit values are returned:  
 12559 0 Successful completion of the listing.  
 12560 >0 An error occurred.  
 12561 Otherwise, the exit status will be that of the commands executed by *fc*.

12562 **CONSEQUENCES OF ERRORS**  
 12563 Default.

12564 **APPLICATION USAGE**  
 12565 Since editors sometimes use file descriptors as integral parts of their editing, redirecting their file  
 12566 descriptors as part of the *fc* command can produce unexpected results. For example, if *vi* is the  
 12567 *FCEDIT* editor, the command:  
 12568 `fc -s | more`  
 12569 will not work correctly on many systems.



12570 Users on windowing systems may want to have separate history files for each window by  
12571 setting *HISTFILE* as follows:

12572 HISTFILE=\$HOME/.sh\_hist\$\$

12573 **EXAMPLES**

12574 None.

12575 **FUTURE DIRECTIONS**

12576 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
12577 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
12578 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
12579 finalised.

12580 **SEE ALSO**

12581 *sh*.

12582 **CHANGE HISTORY**

12583 First released in Issue 4.

12584 **Issue 5**

12585 FUTURE DIRECTIONS section added.

12586 **NAME**

12587 fg — run jobs in the foreground

12588 **SYNOPSIS**12589 JC fg [*job\_id*]12590 **DESCRIPTION**12591 If job control is enabled (see the description of *set -m*), the *fg* utility will move a background job  
12592 from the current environment (see Section 2.12 on page 63) into the foreground.12593 Using *fg* to place a job into the foreground will remove its process ID from the list of those  
12594 “known in the current shell execution environment”; see Section 2.9.3 on page 50.12595 **OPTIONS**

12596 None.

12597 **OPERANDS**

12598 The following operand is supported:

12599 *job\_id* Specify the job to be run as a foreground job. If no *job\_id* operand is given, the *job\_id* for  
12600 the job that was most recently suspended, placed in the background or run as a  
12601 background job will be used. The format of *job\_id* is described in the entry for **job**  
12602 **control job ID** in the XBD specification, **Chapter 2, Glossary**.12603 **STDIN**

12604 Not used.

12605 **INPUT FILES**

12606 None.

12607 **ENVIRONMENT VARIABLES**12608 The following environment variables affect the execution of *fg*:12609 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
12610 *LANG* is unset or null, the corresponding value from the implementation-dependent  
12611 default locale will be used. If any of the internationalisation variables contains an  
12612 invalid setting, the utility will behave as if none of the variables had been defined.12613 *LC\_ALL*12614 If set to a non-empty string value, override the values of all the other  
12615 internationalisation variables.12616 *LC\_CTYPE*12617 Determine the locale for the interpretation of sequences of bytes of text data as  
12618 characters (for example, single- as opposed to multi-byte characters in arguments).12619 *LC\_MESSAGES*12620 Determine the locale that should be used to affect the format and contents of diagnostic  
12621 messages written to standard error.12622 EX *NLSPATH*12623 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.12624 **ASYNCHRONOUS EVENTS**

12625 Default.

12626 **STDOUT**12627 The *fg* utility writes the command line of the job to standard output in the following format:

12628 "%s\n", &lt;command&gt;

12629 **STDERR**

12630           Used only for diagnostic messages.

12631 **OUTPUT FILES**

12632           None.

12633 **EXTENDED DESCRIPTION**

12634           None.

12635 **EXIT STATUS**

12636           The following exit values are returned:

12637           0   Successful completion.

12638           >0  An error occurred.

12639 **CONSEQUENCES OF ERRORS**

12640           If job control is disabled, the *fg* utility will exit with an error and no job will be placed in the  
12641           foreground.

12642 **APPLICATION USAGE**

12643           The *fg* utility will not work as expected when it is operating in its own utility execution  
12644           environment because that environment will have no applicable jobs to manipulate. See the  
12645           APPLICATION USAGE section for *bg*. For this reason, *fg* is generally implemented as a shell  
12646           regular built-in.

12647 **EXAMPLES**

12648           None.

12649 **FUTURE DIRECTIONS**

12650           None.

12651 **SEE ALSO**

12652           *bg*, *kill*, *jobs*, *wait*.

12653 **CHANGE HISTORY**

12654           First released in Issue 4.

12655 **NAME**12656 fgrep — search a file for a fixed-string pattern (**LEGACY**)12657 **SYNOPSIS**12658 OB fgrep [ -c | -l ][-invx] -e *pattern\_list* [*file...*]12659 OB fgrep [ -c | -l ][-invx] -f *pattern\_list* [*file...*]12660 OB fgrep [ -c | -l ][-invx] *pattern\_list* [*file...*]12661 **DESCRIPTION**12662 The name *fgrep* is an obsolescent version equivalent to *grep -F*.12663 A command invoking the *fgrep* utility with the *-e* option specified is equivalent to the command:12664 grep -F [ -c | -l ][-invx] -e *pattern\_list* [*file...*]12665 A command invoking the *fgrep* utility with the *-f* option specified is equivalent to the command:12666 grep -F [ -c | -l ][-invx] -f *pattern\_list* [*file...*]12667 A command invoking the *fgrep* utility with neither the *-e* nor the *-f* option operand specified is  
12668 equivalent to the command:12669 grep -F [ -c | -l ][-invx] *pattern\_list* [*file...*]12670 **OPTIONS**12671 Refer to *grep*.12672 **OPERANDS**12673 Refer to *grep*.12674 **STDIN**12675 Refer to *grep*.12676 **INPUT FILES**12677 Refer to *grep*.12678 **ENVIRONMENT VARIABLES**12679 Refer to *grep*.12680 **ASYNCHRONOUS EVENTS**12681 Refer to *grep*.12682 **STDOUT**12683 Refer to *grep*.12684 **STDERR**12685 Refer to *grep*.12686 **OUTPUT FILES**12687 Refer to *grep*.12688 **EXTENDED DESCRIPTION**12689 Refer to *grep*.12690 **EXIT STATUS**12691 Refer to *grep*.12692 **CONSEQUENCES OF ERRORS**12693 Refer to *grep*.

12694 **APPLICATION USAGE**

12695       Unlike *grep -F*, multiple *-e* or *-f* options produce undefined results. Adjacent newline  
12696       characters in the *pattern* operand or *-e pattern\_list* option-argument also produce undefined  
12697       results.

12698       Applications should migrate to the *grep -F* command.

12699 **EXAMPLES**

12700       Refer to *grep*.

12701 **FUTURE DIRECTIONS**

12702       Refer to *grep*.

12703 **SEE ALSO**

12704       *grep*.

12705 **CHANGE HISTORY**

12706       First released in Issue 2.

12707 **Issue 4**

12708       Aligned with the ISO/IEC 9945-2: 1993 standard.

12709       Separated from the *egrep* description.

12710 **Issue 5**

12711       Marked LEGACY.

12712 **NAME**

12713 file — determine file type

12714 **SYNOPSIS**12715 *file file ...*12716 **DESCRIPTION**12717 The *file* utility performs a series of tests on each specified *file* in an attempt to classify it:

- 12718 1. If the file is not a regular file, its file type is identified. The file types directory, FIFO, block  
12719 special and character special are identified as such. Other implementation-dependent file  
12720 types may also be identified.
- 12721 2. If the file is a regular file, and:
  - 12722 a. The file is zero-length, it is identified as an empty file.
  - 12723 b. The file is not zero-length, *file* will examine an initial segment of the file and make a  
12724 guess at identifying its contents or whether it is an executable binary file. (The  
12725 answer is not guaranteed to be correct.)

12726 If *file* does not exist, cannot be read, or its file status could not be determined, the output will  
12727 indicate that the file was processed, but that its type could not be determined.

12728 **OPTIONS**

12729 None.

12730 **OPERANDS**

12731 The following operand is supported:

12732 *file* A pathname of a file to be tested.12733 **STDIN**

12734 Not used.

12735 **INPUT FILES**12736 The *file* can be any file type.12737 **ENVIRONMENT VARIABLES**12738 The following environment variables affect the execution of *file*:

12739 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
12740 **LANG** is unset or null, the corresponding value from the implementation-dependent  
12741 default locale will be used. If any of the internationalisation variables contains an  
12742 invalid setting, the utility will behave as if none of the variables had been defined.

12743 **LC\_ALL**

12744 If set to a non-empty string value, override the values of all the other  
12745 internationalisation variables.

12746 **LC\_CTYPE**

12747 Determine the locale for the interpretation of sequences of bytes of text data as  
12748 characters (for example, single- as opposed to multi-byte characters in arguments and  
12749 input files).

12750 **LC\_MESSAGES**

12751 Determine the locale that should be used to affect the format and contents of diagnostic  
12752 messages written to standard error and informative messages written to standard  
12753 output.

12754 EX **NLSPATH**  
 12755 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 12756 ASYNCHRONOUS EVENTS

12757 Default.

## 12758 STDOUT

12759 In the POSIX locale, the following format is used to identify each operand, *file* specified:

12760 "%s: %s\n", <file>, <type>

12761 The values for <type> are unspecified, except that in the POSIX locale, if *file* is identified as one  
 12762 of the types listed in the following table, <type> will contain (but is not limited to) the  
 12763 corresponding string. Each space shown in the strings is exactly one space character.

|       | If file is a:                                                   | <type> Contains the String: |
|-------|-----------------------------------------------------------------|-----------------------------|
| 12766 | directory                                                       | directory                   |
| 12767 | FIFO                                                            | fifo                        |
| 12768 | block special                                                   | block special               |
| 12769 | character special                                               | character special           |
| 12770 | executable binary                                               | executable                  |
| 12771 | empty regular file                                              | empty                       |
| 12772 | <i>ar</i> archive library (see <i>ar</i> )                      | archive                     |
| 12773 | extended <i>cpio</i> format (see <i>pax</i> )                   | cpio archive                |
| 12774 | extended <i>tar</i> format (see <i>ustar</i> — see <i>pax</i> ) | tar archive                 |
| 12775 | shell script                                                    | commands text               |
| 12776 | C-language source                                               | c program text              |
| 12777 | FORTRAN source                                                  | fortran program text        |

12778 **Table 3-6** File Utility Output Strings

12779 If the file named by the *file* operand does not exist, cannot be read or the status of the file cannot  
 12780 be determined, the string **cannot open** will be included as part of the <type> field, but this is not  
 12781 considered an error that affects the exit status.

## 12782 STDERR

12783 Used only for diagnostic messages.

## 12784 OUTPUT FILES

12785 None.

## 12786 EXTENDED DESCRIPTION

12787 None.

## 12788 EXIT STATUS

12789 The following exit values are returned:

12790 0 Successful completion.

12791 >0 An error occurred.

## 12792 CONSEQUENCES OF ERRORS

12793 Default.

12794 **APPLICATION USAGE**

12795       The *file* utility can only be required to guess at many of the file types because only exhaustive  
12796       testing can determine some types with certitude. For example, binary data on some systems  
12797       might match the initial segment of an executable or a *tar* archive.

12798       Note that the table indicates that the output contains the stated string. Systems may add text  
12799       before or after the string. For executables, as an example, the machine architecture and various  
12800       facts about how the file was link-edited may be included.

12801 **EXAMPLES**

12802       Determine if an argument is a binary executable file:

```
12803 file "$1" | grep -Fq executable &&
12804 printf "%s is executable.\n" "$1"
```

12805 **FUTURE DIRECTIONS**

12806       None.

12807 **SEE ALSO**

12808       *ls*.

12809 **CHANGE HISTORY**

12810       First released in Issue 4.



12811 **NAME**12812        **find** — find files12813 **SYNOPSIS**12814        **find** *path...* [*operand\_expression*]12815 **DESCRIPTION**

12816        The *find* utility will recursively descend the directory hierarchy from each file specified by *path*,  
 12817        evaluating a Boolean expression composed of the primaries described in the OPERANDS section  
 12818        for each file encountered.

12819        The *find* utility will be able to descend to arbitrary depths in a file hierarchy and will not fail due  
 12820        to path length limitations (unless a *path* operand specified by the application exceeds  
 12821        {PATH\_MAX} requirements).

12822 **OPTIONS**

12823        None.

12824 **OPERANDS**

12825        The following operands are supported:

12826        The *path* operand is a pathname of a starting point in the directory hierarchy.

12827        The first argument that starts with a "-", or is a "!" or a "(", and all subsequent arguments will be  
 12828        interpreted as an *expression* made up of the following primaries and operators. In the  
 12829        descriptions, wherever *n* is used as a primary argument, it will be interpreted as a decimal  
 12830        integer optionally preceded by a plus (+) or minus (-) sign, as follows:

12831        **+n**        more than *n*12832        **n**        exactly *n*12833        **-n**        less than *n*.

12834        The following primaries are supported:

12835        **-name** *pattern*

12836        The primary will evaluate as true if the basename of the filename being examined  
 12837        matches *pattern* using the pattern matching notation described in Section 2.13 on page  
 12838        64.

12839        **-nouser** The primary will evaluate as true if the file belongs to a user ID for which the **XSH**  
 12840        specification *getpwuid()* (or equivalent) function returns NULL.

12841        **-nogroup**

12842        The primary will evaluate as true if the file belongs to a group ID for which the **XSH**  
 12843        specification *getgrgid()* (or equivalent) function returns NULL.

12844        **-xdev**    The primary always will evaluate as true; it will cause *find* not to continue descending  
 12845        past directories that have a different device ID (*st\_dev*, see the **XSH** specification *stat()*  
 12846        function). If any **-xdev** primary is specified, it will apply to the entire expression even  
 12847        if the **-xdev** primary would not normally be evaluated.

12848        **-prune** The primary always will evaluate as true; it will cause *find* not to descend the current  
 12849        pathname if it is a directory. If the **-depth** primary is specified, the **-prune** primary  
 12850        will have no effect.

12851        **-perm** [-] *mode*

12852        The *mode* argument is used to represent file mode bits. It will be identical in format to  
 12853        the *symbolic\_mode* operand described in *chmod* on page 193, and will be interpreted as  
 12854        follows. To start, a template will be assumed with all file mode bits cleared. An *op*

12855 symbol of "+" will set the appropriate mode bits in the template; "-" will clear the  
 12856 appropriate bits; "=" will set the appropriate mode bits, without regard to the contents  
 12857 of process' file mode creation mask. The *op* symbol of "-" cannot be the first character  
 12858 of *mode*; this avoids ambiguity with the optional leading hyphen. Since the initial mode  
 12859 is all bits off, there are not any symbolic modes that need to use "-" as the first  
 12860 character.

12861 If the hyphen is omitted, the primary will evaluate as true when the file permission bits  
 12862 exactly match the value of the resulting template.

12863 Otherwise, if *mode* is prefixed by a hyphen, the primary will evaluate as true if at least  
 12864 all the bits in the resulting template are set in the file permission bits.

12865 EX **-perm [-] *onum***  
 12866 If the hyphen is omitted, the primary will evaluate as true when the file permission bits  
 12867 exactly match the value of the octal number *onum* and only the bits corresponding to  
 12868 the octal mask 07777 will be compared. (See the description of the octal *mode* in *chmod*.)  
 12869 Otherwise, if *onum* is prefixed by a hyphen, the primary will evaluate as true if at least  
 12870 all of the bits specified in *onum* that are also set in the octal mask 07777 are set.

12871 **-type *c*** The primary will evaluate as true if the type of the file is *c*, where *c* is b, c, d, p or f for  
 12872 block special file, character special file, directory, FIFO or regular file, respectively.

12873 **-links *n***  
 12874 The primary will evaluate as true if the file has *n* links.

12875 **-user *uname***  
 12876 The primary will evaluate as true if the file belongs to the user *uname*. If *uname* is a  
 12877 decimal integer and the *getpwnam()* (or equivalent) function does not return a valid  
 12878 user name, *uname* will be interpreted as a user ID.

12879 **-group *gname***  
 12880 The primary will evaluate as true if the file belongs to the group *gname*. If *gname* is a  
 12881 decimal integer and the *getgrnam()* (or equivalent) function does not return a valid  
 12882 group name, *gname* will be interpreted as a group ID.

12883 **-size *n*[*c*]**  
 12884 The primary will evaluate as true if the file size in bytes, divided by 512 and rounded  
 12885 up to the next integer, is *n*. If *n* is followed by the character *c*, the size will be in bytes.

12886 **-atime *n***  
 12887 The primary will evaluate as true if the file access time subtracted from the  
 12888 initialisation time is *n*-1 to *n* multiples of 24 hours. The initialisation time will be a  
 12889 time between the invocation of the *find* utility and the first access by that invocation of  
 12890 the *find* utility to any file specified by its *path* operands. For example, **-atime 3** is true if  
 12891 the file was accessed any time in the period from 72 to 48 hours ago.

12892 **-mtime *n***  
 12893 The primary will evaluate as true if the file modification time subtracted from the  
 12894 initialisation time is *n*-1 to *n* multiples of 24 hours. The initialisation time will be a  
 12895 time between the invocation of the *find* utility and the first access by that invocation of  
 12896 the *find* utility to any file specified by its *path* operands.

12897 **-ctime *n***  
 12898 The primary will evaluate as true if the time of last change of file status information  
 12899 subtracted from the initialisation time is *n*-1 to *n* multiples of 24 hours. The  
 12900 initialisation time will be a time between the invocation of the *find* utility and the first  
 12901 access by that invocation of the *find* utility to any file specified by its *path* operands.

12902       **-exec** *utility\_name* [*argument* ...] ;  
 12903           The primary will evaluate as true if the invoked utility *utility\_name* returns a zero value  
 12904           as exit status. The end of the primary expression will be punctuated by a semicolon. A  
 12905           *utility\_name* or *argument* containing only the two characters {} will be replaced by the  
 12906           current pathname. If a *utility\_name* or argument string contains the two characters {},  
 12907           but not just the two characters {}, it is implementation-dependent whether *find* replaces  
 12908           those two characters with the current pathname or uses the string without change. The  
 12909           current directory for the invocation of *utility\_name* will be the same as the current  
 12910           directory when the *find* utility was started. If the *utility\_name* names any of the special  
 12911           built-in utilities in Section 2.14 on page 67, the results are undefined.

12912       **-ok** *utility\_name* [*argument* ...] ;  
 12913           The **-ok** primary will be equivalent to **-exec**, except that *find* will request affirmation of  
 12914           the invocation of *utility\_name* using the current file as an argument by writing to  
 12915           standard error as described in the STDERR section. If the response on standard input is  
 12916           affirmative, the utility will be invoked. Otherwise, the command will not be invoked  
 12917           and the value of the **-ok** operand will be false.

12918       **-print**   The primary always will evaluate as true; it will cause the current pathname to be  
 12919           written to standard output.

12920       **-newer** *file*  
 12921           The primary will evaluate as true if the modification time of the current file is more  
 12922           recent than the modification time of the file named by the pathname *file*.

12923       **-depth**   The primary always will evaluate as true; it will cause descent of the directory  
 12924           hierarchy to be done so that all entries in a directory are acted on before the directory  
 12925           itself. If a **-depth** primary is not specified, all entries in a directory will be acted on  
 12926           after the directory itself. If any **-depth** primary is specified, it will apply to the entire  
 12927           expression even if the **-depth** primary would not normally be evaluated.

12928       The primaries can be combined using the following operators (in order of decreasing  
 12929       precedence):

12930       ( *expression* )  
 12931           True if *expression* is true.

12932       ! *expression*  
 12933           Negation of a primary; the unary NOT operator.

12934       *expression* [**-a**] *expression*  
 12935           Conjunction of primaries; the AND operator will be implied by the juxtaposition of two  
 12936           primaries or made explicit by the optional **-a** operator. The second expression will not  
 12937           be evaluated if the first expression is false.

12938       *expression* **-o** *expression*  
 12939           Alternation of primaries; the OR operator. The second expression will not be evaluated  
 12940           if the first expression is true.

12941       If no *expression* is present, **-print** will be used as the expression. Otherwise, if the given  
 12942       expression does not contain any of the primaries **-exec**, **-ok** or **-print**, the given expression will  
 12943       be effectively replaced by:

12944       ( *given\_expression* ) -print

12945       The **-user**, **-group** and **-newer** primaries each will evaluate their respective arguments only  
 12946       once.

12947 **STDIN**

12948 If the **–ok** primary is used, the response will be read from the standard input. An entire line will  
 12949 be read as the response. Otherwise, the standard input will not be used.

12950 **INPUT FILES**

12951 None.

12952 **ENVIRONMENT VARIABLES**

12953 The following environment variables affect the execution of *find*:

12954 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 12955 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 12956 default locale will be used. If any of the internationalisation variables contains an  
 12957 invalid setting, the utility will behave as if none of the variables had been defined.

12958 **LC\_ALL**

12959 If set to a non-empty string value, override the values of all the other  
 12960 internationalisation variables.

12961 **LC\_COLLATE**

12962 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 12963 character collating elements used in the pattern matching notation for the **–n** option  
 12964 and in the extended regular expression defined for the **yesexpr** locale keyword in the  
 12965 **LC\_MESSAGES** category.

12966 **LC\_CTYPE**

12967 This variable will determine the locale for the interpretation of sequences of bytes of  
 12968 text data as characters (for example, single- versus multi-byte characters in arguments),  
 12969 the behaviour of character classes within the pattern matching notation used for the **–n**  
 12970 option, and the behaviour of character classes within regular expressions used in the  
 12971 extended regular expression defined for the **yesexpr** locale keyword in the  
 12972 **LC\_MESSAGES** category.

12973 **LC\_MESSAGES**

12974 Determine the locale for the processing of affirmative responses that should be used to  
 12975 affect the format and contents of diagnostic messages written to standard error.

12976 EX **NLSPATH**

12977 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

12978 **PATH** Determine the location of the *utility\_name* for the **–exec** and **–ok** primaries, as described  
 12979 in the **XBD** specification, **Chapter 6, Environment Variables**.

12980 **ASYNCHRONOUS EVENTS**

12981 Default.

12982 **STDOUT**

12983 The **–print** primary will cause the current pathnames to be written to standard output. The  
 12984 format will be:

12985 "%s\n", <path>

12986 **STDERR**

12987 The **–ok** primary will write a prompt to standard error containing at least the *utility\_name* to be  
 12988 invoked and the current pathname. In the POSIX locale, the last non-blank character in the  
 12989 prompt will be "?". The exact format used is unspecified.

12990 Otherwise, the standard error will be used only for diagnostic messages.

12991 **OUTPUT FILES**

12992       None.

12993 **EXTENDED DESCRIPTION**

12994       None.

12995 **EXIT STATUS**

12996       The following exit values are returned:

12997       0   All *path* operands were traversed successfully.

12998       &gt;0  An error occurred.

12999 **CONSEQUENCES OF ERRORS**

13000       Default.

13001 **APPLICATION USAGE**13002       When used in operands, pattern matching notation, semicolons, opening parentheses, and  
13003       closing parentheses are special to the shell and must be quoted (see Section 2.2 on page 20).13004       The bit that is traditionally used for sticky (historically 01000) is still specified in the **-perm**  
13005       primary using the octal number argument form. Since this bit is not defined by this  
13006       specification, applications must not assume that it actually refers to the traditional sticky bit.13007       The references to octal modes are marked EX because, although they are obsolescent in the  
13008       ISO/IEC 9945-2: 1993 standard, The Open Group is committed to maintaining them for portable  
13009       applications until further notice.13010 **EXAMPLES**

13011       1. The following commands are equivalent:

13012            `find .`13013            `find . -print`

13014       They both write out the entire directory hierarchy from the current directory.

13015       2. The following command:

13016            `find / \( -name tmp -o -name '*.xx' \) -atime +7 -exec rm {} \;`13017       removes all files named **tmp** or ending in **.xx** that have not been accessed for seven or more  
13018       24-hour periods.

13019       3. The following command:

13020            `find . -perm -o+w,+s`13021       prints (**-print** is assumed) the names of all files in or below the current directory, with all  
13022       of the file permission bits **S\_ISUID**, **S\_ISGID** and **S\_IWOTH** set.

13023       4. The following command:

13024            `find . -name SCCS -prune -o -print`13025       recursively prints pathnames of all files in the current directory and below, but skips  
13026       directories named **SCCS** and files in them.

13027       5. The following command:

13028            `find . -print -name SCCS -prune`13029       behaves as in the previous example, but prints the names of the **SCCS** directories.

13030 6. The following command is roughly equivalent to the **-nt** extension to *test*:

```
13031 if [-n "$(find file1 -prune -newer file2)"]; then
13032 printf %s\\n "file1 is newer than file2"
13033 fi
```

13034 7. The descriptions of **-atime**, **-ctime** and **-mtime** use the terminology *n* “24-hour periods”.  
 13035 For example, a file accessed at 23:59 will be selected by:

```
13036 find . -atime -1 -print
```

13037 at 00:01 the next day (less than 24 hours later, not more than one day ago); the midnight  
 13038 boundary between days has no effect on the 24-hour calculation.

#### 13039 FUTURE DIRECTIONS

13040 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 13041 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 13042 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 13043 finalised.

#### 13044 SEE ALSO

13045 *chmod*, *pax*, *sh*, *test*, the **XSH** specification description of *stat()*.

#### 13046 CHANGE HISTORY

13047 First released in Issue 2.

#### 13048 Issue 4

13049 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 13050 Issue 5

13051 FUTURE DIRECTIONS section added.

13052 **NAME**

13053 fold — filter for folding lines

13054 **SYNOPSIS**13055 fold [-bs][-w *width*][*file...*]13056 **DESCRIPTION**

13057 The *fold* utility is a filter that will fold lines from its input files, breaking the lines to have a  
 13058 maximum of *width* column positions (or bytes, if the **-b** option is specified). Lines will be broken  
 13059 by the insertion of a newline character such that each output line (referred to later in this section  
 13060 as a segment) is the maximum width possible that does not exceed the specified number of  
 13061 column positions (or bytes). A line will not be broken in the middle of a character. The  
 13062 behaviour is undefined if *width* is less than the number of columns any single character in the  
 13063 input would occupy.

13064 If the carriage-return, backspace or tab characters are encountered in the input, and the **-b**  
 13065 option is not specified, they will be treated specially:

13066 backspace

13067 The current count of line width will be decremented by one, although the count never  
 13068 will become negative. The *fold* utility will not insert a newline character immediately  
 13069 before or after any backspace character.

13070 carriage-return

13071 The current count of line width will be set to zero. The *fold* utility will not insert a  
 13072 newline character immediately before or after any carriage-return character.

13073 tab Each tab character encountered will advance the column position pointer to the next  
 13074 tab stop. Tab stops will be at each column position *n* such that *n* modulo 8 equals 1.

13075 **OPTIONS**13076 The *fold* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

13077 The following options are supported:

13078 **-b** Count *width* in bytes rather than column positions.

13079 **-s** If a segment of a line contains a blank character within the first *width* column positions  
 13080 (or bytes), break the line after the last such blank character meeting the width  
 13081 constraints. If there is no blank character meeting the requirements, the **-s** option will  
 13082 have no effect for that output segment of the input line.

13083 **-w *width***

13084 Specify the maximum line length, in column positions (or bytes if **-b** is specified). The  
 13085 results are unspecified if *width* is not a positive decimal number. The default value is  
 13086 80.

13087 **OPERANDS**

13088 The following operand is supported:

13089 *file* A pathname of a text file to be folded. If no *file* operands are specified, the standard  
 13090 input will be used.

13091 **STDIN**

13092 The standard input will be used only if no *file* operands are specified. See the INPUT FILES  
 13093 section.

13094 **INPUT FILES**

13095 If the **-b** option is specified, the input files must be text files except that the lines are not limited  
 13096 to {LINE\_MAX} bytes in length. If the **-b** option is not specified, the input files must be text

13097 files.

## 13098 ENVIRONMENT VARIABLES

13099 The following environment variables affect the execution of *fold*:

13100 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 13101 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 13102 default locale will be used. If any of the internationalisation variables contains an  
 13103 invalid setting, the utility will behave as if none of the variables had been defined.

13104 **LC\_ALL**

13105 If set to a non-empty string value, override the values of all the other  
 13106 internationalisation variables.

13107 **LC\_CTYPE**

13108 Determine the locale for the interpretation of sequences of bytes of text data as  
 13109 characters (for example, single- as opposed to multi-byte characters in arguments and  
 13110 input files), and for the determination of the width in column positions each character  
 13111 would occupy on a constant-width font output device.

13112 **LC\_MESSAGES**

13113 Determine the locale that should be used to affect the format and contents of diagnostic  
 13114 messages written to standard error.

13115 EX **NLSPATH**

13116 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 13117 ASYNCHRONOUS EVENTS

13118 Default.

## 13119 STDOUT

13120 The standard output will be a file containing a sequence of characters whose order will be  
 13121 preserved from the input files, possibly with inserted newline characters.

## 13122 STDERR

13123 Used only for diagnostic messages.

## 13124 OUTPUT FILES

13125 None.

## 13126 EXTENDED DESCRIPTION

13127 None.

## 13128 EXIT STATUS

13129 The following exit values are returned:

13130 0 All input files were processed successfully.

13131 >0 An error occurred.

## 13132 CONSEQUENCES OF ERRORS

13133 Default.

## 13134 APPLICATION USAGE

13135 The *cut* and *fold* utilities can be used to create text files out of files with arbitrary line lengths.

13136 The *cut* utility should be used when the number of lines (or records) needs to remain constant.

13137 The *fold* utility should be used when the contents of long lines need to be kept contiguous.

13138 The *fold* utility is frequently used to send text files to printers that truncate, rather than fold, lines  
 13139 wider than the printer is able to print (usually 80 or 132 column positions).



13140 **EXAMPLES**

13141       An example invocation that submits a file of possibly long lines to the printer (under the  
13142       assumption that the user knows the line width of the printer to be assigned by *lp*):

13143       fold -w 132 bigfile | lp

13144 **FUTURE DIRECTIONS**

13145       None.

13146 **SEE ALSO**

13147       *cut*.

13148 **CHANGE HISTORY**

13149       First released in Issue 4.

## 13150 NAME

13151 fort77 — FORTRAN compiler (**FORTRAN**)

## 13152 SYNOPSIS

13153 fort77 [-c][-g][-L *directory*]... [-O *optlevel*][-o *outfile*][-s][-w]  
 13154 *operand*...

## 13155 DESCRIPTION

13156 The *fort77* utility is the interface to the FORTRAN compilation system; it will accept the full  
 13157 FORTRAN-77 language defined by the ANSI X3.9-1978 standard. The system conceptually  
 13158 consists of a compiler and link editor. The files referenced by *operands* are compiled and linked  
 13159 to produce an executable file. It is unspecified whether the linking occurs entirely within the  
 13160 operation of *fort77*; some systems may produce objects that are not fully resolved until the file is  
 13161 executed.

13162 If the **-c** option is present, for all pathname operands of the form *file.f*, the files:

13163 \$(basename *pathname* .f).o

13164 will be created or overwritten as the result of successful compilation. If the **-c** option is not  
 13165 specified, it is unspecified whether such .o files are created or deleted for the *file.f* operands.

13166 If there are no options that prevent link editing (such as **-c**) and all operands compile and link  
 13167 without error, the resulting executable file will be written into the file named by the **-o** option (if  
 13168 present) or to the file **a.out**. The executable file will be created as specified in the **XSH**  
 13169 specification, except that the file permissions will be set to:

13170 S\_IRWXO | S\_IRWXG | S\_IRWXU

13171 and that the bits specified by the *umask* of the process will be cleared.

## 13172 OPTIONS

13173 The *fort77* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**,  
 13174 except that:

- 13175 • The **-l** *library* operands have the format of options, but their position within a list of  
 13176 operands affects the order in which libraries are searched.
- 13177 • The order of specifying the multiple **-L** options is significant.
- 13178 • Portable applications must specify each option separately; that is, grouping option letters (for  
 13179 example, **-cg**) need not be recognised by all implementations.

13180 The following options are supported:

- 13181 **-c** Suppress the link-edit phase of the compilation, and do not remove any object files that  
 13182 are produced.
- 13183 **-g** Produce symbolic information in the object or executable files; the nature of this  
 13184 information is unspecified, and may be modified by implementation-dependent  
 13185 interactions with other options.
- 13186 **-s** Produce object or executable files, or both, from which symbolic and other information  
 13187 not required for proper execution using the **XSH** specification *exec* family has been  
 13188 removed (stripped). If both **-g** and **-s** options are present, the action taken is  
 13189 unspecified.

13190       **-o** *outfile*  
 13191           Use the pathname *outfile*, instead of the default **a.out**, for the executable file produced.  
 13192           If the **-o** option is present with **-c**, the result is unspecified.

13193       **-L** *directory*  
 13194           Change the algorithm of searching for the libraries named in **-l** operands to look in the  
 13195           directory named by the *directory* pathname before looking in the usual places.  
 13196           Directories named in **-L** options will be searched in the specified order. At least ten  
 13197           instances of this option will be supported in a single *fort77* command invocation. If a  
 13198           directory specified by a **-L** option contains a file named **libf.a**, the results are  
 13199           unspecified.

13200       **-O** *optlevel*  
 13201           Specify the level of code optimisation. If the *optlevel* option-argument is the digit 0, all  
 13202           special code optimisations will be disabled. If it is the digit 1, the nature of the  
 13203           optimisation is unspecified. If the **-O** option is omitted, the nature of the system's  
 13204           default optimisation is unspecified. It is unspecified whether code generated in the  
 13205           presence of the **-O 0** option is the same as that generated when **-O** is omitted. Other  
 13206           *optlevel* values may be supported.

13207       **-w**       Suppress warnings.

13208       Multiple instances of **-L** options can be specified.

13209 **OPERANDS**

13210       An *operand* is either in the form of a pathname or the form **-l library**. At least one operand of the  
 13211       pathname form will be specified. The following operands are supported:

13212       *file.f*     The pathname of a FORTRAN source file to be compiled and optionally passed to the  
 13213                   link editor. The filename operand will be of this form if the **-c** option is used.

13214       *file.a*     A library of object files typically produced by *ar*, and passed directly to the link editor.  
 13215                   Implementations may recognise implementation-dependent suffixes other than **.a** as  
 13216                   denoting object file libraries.

13217       *file.o*     An object file produced by *fort77 -c* and passed directly to the link editor.  
 13218                   Implementations may recognise implementation-dependent suffixes other than **.o** as  
 13219                   denoting object files.

13220       The processing of other files is implementation-dependent.

13221       **-l library**  
 13222           (The letter ell.) Search the library named:

13223                *liblibrary.a*

13224       A library is searched when its name is encountered, so the placement of a **-l** operand is  
 13225       significant. Several standard libraries can be specified in this manner, as described in  
 13226       the EXTENDED DESCRIPTION section. Implementations may recognise  
 13227       implementation-dependent suffixes other than **.a** as denoting libraries.

13228 **STDIN**

13229       Not used.

13230 **INPUT FILES**

13231       The input file must be one of the following: a text file containing FORTRAN source code; an  
 13232       object file in the format produced by *fort77 -c*; or a library of object files, in the format produced  
 13233       by archiving zero or more object files, using *ar*. Implementations may supply additional utilities  
 13234       that produce files in these formats. Additional input files are implementation-dependent.

13235 A tab character encountered within the first six characters on a line of source code will cause the  
 13236 compiler to interpret the following character as if it were the seventh character on the line (that  
 13237 is, in column 7).

#### 13238 ENVIRONMENT VARIABLES

13239 The following environment variables affect the execution of *fort77*:

13240 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 13241 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 13242 default locale will be used. If any of the internationalisation variables contains an  
 13243 invalid setting, the utility will behave as if none of the variables had been defined.

13244 **LC\_ALL**

13245 If set to a non-empty string value, override the values of all the other  
 13246 internationalisation variables.

13247 **LC\_CTYPE**

13248 Determine the locale for the interpretation of sequences of bytes of text data as  
 13249 characters (for example, single- as opposed to multi-byte characters in arguments and  
 13250 input files).

13251 **LC\_MESSAGES**

13252 Determine the locale that should be used to affect the format and contents of diagnostic  
 13253 messages written to standard error.

13254 EX **NLSPATH**

13255 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

13256 **TMPDIR**

13257 Determine the pathname that should override the default directory for temporary files,  
 13258 if any.

#### 13259 ASYNCHRONOUS EVENTS

13260 Default.

#### 13261 STDOUT

13262 Not used.

#### 13263 STDERR

13264 Used only for diagnostic messages. If more than one file operand ending in *.f* (or possibly other  
 13265 unspecified suffixes) is given, for each such file:

13266 " %s:\n", <file>

13267 may be written to allow identification of the diagnostic message with the appropriate input file.

13268 This utility may produce warning messages about certain conditions that do not warrant  
 13269 returning an error (non-zero) exit value.

#### 13270 OUTPUT FILES

13271 Object files, listing files and executable files are produced in unspecified formats.

13272 **EXTENDED DESCRIPTION**13273 **Standard Libraries**

13274 The *fort77* utility recognises the following **-l** operand for the standard library:

13275 **-l f** This library contains all library functions referenced in the ANSI X3.9-1978 standard.  
13276 This operand is not required to be present to cause a search of this library.

13277 In the absence of options that inhibit invocation of the link editor, such as **-c**, the *fort77* utility  
13278 will cause the equivalent of a **-l f** operand to be passed to the link editor as the last **-l** operand,  
13279 causing it to be searched after all other object files and libraries are loaded.

13280 It is unspecified whether the library **libf.a** exists as a regular file. The implementation may  
13281 accept as **-l** operands names of objects that do not exist as regular files.

13282 **External Symbols**

13283 The FORTRAN compiler and link editor support the significance of external symbols up to a  
13284 length of at least 31 bytes; case folding is permitted. The action taken upon encountering  
13285 symbols exceeding the implementation-dependent maximum symbol length is unspecified.

13286 The compiler and link editor support a minimum of 511 external symbols per source or object  
13287 file, and a minimum of 4095 external symbols total. A diagnostic message is written to standard  
13288 output if the implementation-dependent limit is exceeded; other actions are unspecified.

13289 **EXIT STATUS**

13290 The following exit values are returned:

13291 0 Successful compilation or link edit.  
13292 >0 An error occurred.

13293 **CONSEQUENCES OF ERRORS**

13294 When *fort77* encounters a compilation error, it will write a diagnostic to standard error and  
13295 continue to compile other source code operands. It will return a non-zero exit status, but it is  
13296 implementation-dependent whether an object module is created. If the link edit is unsuccessful,  
13297 a diagnostic message will be written to standard error, and *fort77* will exit with a non-zero  
13298 status.

13299 **APPLICATION USAGE**

13300 None.

13301 **EXAMPLES**

13302 The following are examples of usage:

13303 *fort77* -o foo xyz.f  
13304 Compiles **xyz.f** and creates the executable file **foo**.

13305 *fort77* -c xyz.f  
13306 Compiles **xyz.f** and creates the object file **xyz.o**.

13307 *fort77* xyz.f  
13308 Compiles **xyz.f** and creates the executable file **a.out**.

13309 *fort77* xyz.f b.o  
13310 Compiles **xyz.f**, links it with **b.o** and creates the executable **a.out**.

13311 **FUTURE DIRECTIONS**

13312 A compilation system based on FORTRAN-90 (ISO/IEC 1539:1991) will be considered for a  
13313 future issue; it may have a different utility name from *fort77*.

13314 **SEE ALSO**

13315           *ar, asa, c89, umask.*

13316 **CHANGE HISTORY**

13317           First released in Issue 4.

|

13318 **NAME**

13319           fuser — list process IDs of all processes that have one or more files open

13320 **SYNOPSIS**

13321 EX       fuser [ -cfu ] *file* ...

13322

13323 **DESCRIPTION**

13324           The *fuser* utility writes to standard output the process IDs of processes running on the local  
13325           system that have one or more named files open. For block special devices, all processes using  
13326           any file on that device are listed.

13327           The *fuser* utility writes to standard error additional information about the named files indicating  
13328           how the file is being used.

13329           Any output for processes running on remote systems that have a named file open is unspecified.

13330           A user may need appropriate privilege to invoke the *fuser* utility.

13331 **OPTIONS**

13332           The *fuser* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

13333           The following options are supported:

13334           **-c**       The file is treated as a mount point and the utility reports on any files open in the file  
13335           system.

13336           **-f**       The report is only for the named files.

13337           **-u**       The user name, in parentheses, associated with each process ID written to standard  
13338           output is written to standard error.

13339 **OPERANDS**

13340           *file*     A pathname on which the file or file system is to be reported.

13341 **STDIN**

13342           Not used.

13343 **INPUT FILES**

13344           The user database.

13345 **ENVIRONMENT VARIABLES**

13346           The following environment variables affect the execution of *fuser*:

13347           **LANG**    Provide a default value for the internationalization variables that are unset or null. If  
13348           **LANG** is unset or null, the corresponding value from the implementation-dependent  
13349           default locale will be used. If any of the internationalization variables contain an  
13350           invalid setting, the utility will behave as if none of the variables had been set.

13351           **LC\_ALL** If set to a non-empty string value, override the values of all the other  
13352           internationalization variables.

13353           **LC\_CTYPE**

13354           Determine the locale for the interpretation of sequences of bytes of text data as  
13355           characters (for example, single- as opposed to multi-byte characters in arguments).

13356           **LC\_MESSAGES**

13357           Determine the locale that should be used to affect the format and contents of diagnostic  
13358           messages written to standard error.

|       |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 13359 | <i>NLSPATH</i>                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 13360 | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                  |  |
| 13361 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 13362 | Default.                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 13363 | <b>STDOUT</b>                                                                                                                                                                                                                                                                                                                                                                                                            |  |
| 13364 | The <i>fuser</i> utility will write the process ID for each process using each file given as an operand to standard output in the following format:                                                                                                                                                                                                                                                                      |  |
| 13365 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13366 | "%d", <process_id>                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 13367 | <b>STDERR</b>                                                                                                                                                                                                                                                                                                                                                                                                            |  |
| 13368 | The <i>fuser</i> utility writes diagnostic messages to standard error.                                                                                                                                                                                                                                                                                                                                                   |  |
| 13369 | The <i>fuser</i> utility also writes the following to standard error:                                                                                                                                                                                                                                                                                                                                                    |  |
| 13370 | <ul style="list-style-type: none"> <li>• The pathname of each named file is written followed immediately by a colon.</li> </ul>                                                                                                                                                                                                                                                                                          |  |
| 13371 | <ul style="list-style-type: none"> <li>• For each process ID written to standard output, the character <i>c</i> is written to standard error if the process is using the file as its current directory and the character <i>r</i> is written to standard error if the process is using the file as its root directory. Implementations may write other alphabetic characters to indicate other uses of files.</li> </ul> |  |
| 13372 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13373 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13374 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13375 | <ul style="list-style-type: none"> <li>• When the <i>-u</i> option is specified, characters indicating the use of the file are followed immediately by the user name, in parentheses, corresponding to the process' real user ID. If the user name cannot be resolved from the process' real user ID, the process' real user ID is written instead of the user name.</li> </ul>                                          |  |
| 13376 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13377 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13378 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13379 | When standard output and standard error are directed to the same file, the output is interspersed so that the filename appears at the start of each line, followed by the process ID and characters indicating the use of the file. Then, if the <i>-u</i> option is specified, the user name or user ID for each process using that file is written.                                                                    |  |
| 13380 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13381 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13382 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13383 | A new line character is written to standard error after the last output described above for each file operand.                                                                                                                                                                                                                                                                                                           |  |
| 13384 |                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13385 | <b>OUTPUT FILES</b>                                                                                                                                                                                                                                                                                                                                                                                                      |  |
| 13386 | None.                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 13387 | <b>EXTENDED DESCRIPTION</b>                                                                                                                                                                                                                                                                                                                                                                                              |  |
| 13388 | None.                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 13389 | <b>EXIT STATUS</b>                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 13390 | The following exit values are returned:                                                                                                                                                                                                                                                                                                                                                                                  |  |
| 13391 | 0 Successful completion.                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 13392 | >0 An error occurred.                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 13393 | <b>CONSEQUENCES OF ERRORS</b>                                                                                                                                                                                                                                                                                                                                                                                            |  |
| 13394 | Default.                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 13395 | <b>APPLICATION USAGE</b>                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 13396 | None.                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 13397 | <b>EXAMPLES</b>                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 13398 | The command:                                                                                                                                                                                                                                                                                                                                                                                                             |  |
| 13399 | <code>fuser -fu .</code>                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 13400 | writes to standard output the process IDs of processes that are using the current directory and                                                                                                                                                                                                                                                                                                                          |  |
| 13401 | writes to standard error an indication of how those processes are using the directory and the                                                                                                                                                                                                                                                                                                                            |  |



13402 user names associated with the processes that are using the current directory.

13403 **FUTURE DIRECTIONS**

13404 None.

13405 **SEE ALSO**

13406 None.

13407 **CHANGE HISTORY**

13408 First released in Issue 5.

13409 **NAME**

13410 gencat — generate a formatted message catalogue

13411 **SYNOPSIS**13412 EX gencat *catfile* *msgfile*...13413 **DESCRIPTION**

13414 The *gencat* utility merges the message text source files *msgfile* into a formatted message  
 13415 catalogue *catfile*. The file *catfile* will be created if it does not already exist. If *catfile* does exist, its  
 13416 messages will be included in the new *catfile*. If set and message numbers collide, the new  
 13417 message text defined in *msgfile* will replace the old message text currently contained in *catfile*.

13418 **OPTIONS**

13419 None.

13420 **OPERANDS**

13421 The following operands are supported:

13422 *catfile* A pathname of the formatted message catalogue. If "-" is specified, standard output is  
 13423 used. The format of the message catalogue produced is unspecified.

13424 *msgfile* A pathname of a message text source file. If "-" is specified for an instance of *msgfile*,  
 13425 standard input is used. The format of message text source files is defined in the  
 13426 EXTENDED DESCRIPTION section.

13427 **STDIN**13428 The standard input is not used unless a *msgfile* operand is specified as "-".13429 **INPUT FILES**

13430 The input files are text files.

13431 **ENVIRONMENT VARIABLES**13432 The following environment variables affect the execution of *gencat*:

13433 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 13434 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 13435 default locale will be used. If any of the internationalisation variables contains an  
 13436 invalid setting, the utility will behave as if none of the variables had been defined.

13437 *LC\_ALL*

13438 If set to a non-empty string value, override the values of all the other  
 13439 internationalisation variables.

13440 *LC\_CTYPE*

13441 Determine the locale for the interpretation of sequences of bytes of text data as  
 13442 characters (for example, single- as opposed to multi-byte characters in arguments and  
 13443 input files).

13444 *LC\_MESSAGES*

13445 Determine the locale that should be used to affect the format and contents of diagnostic  
 13446 messages written to standard error.

13447 *NLSPATH*13448 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.13449 **ASYNCHRONOUS EVENTS**

13450 Default.

13451 **STDOUT**13452 The standard output is not used unless the *catfile* operand is specified as -.

13453 **STDERR**

13454       Used only for diagnostic messages.

13455 **OUTPUT FILES**

13456       None.

13457 **EXTENDED DESCRIPTION**

13458       The format of a message text source file is defined as follows. Note that the fields of a message  
 13459       text source line are separated by a single blank character. Any other blank characters are  
 13460       considered as being part of the subsequent field.

13461       **\$set** *n* *comment*

13462           This line specifies the set identifier of the following messages until the next **\$set** or  
 13463           end-of-file appears. The *n* denotes the set identifier, which is defined as a number in  
 13464           the range [1, {NL\_SETMAX}] (see the **XSH** specification, <limits.h>). Set identifiers  
 13465           must be presented in ascending order within a single source file but need not be  
 13466           contiguous. Any string following the set identifier is treated as a comment. If no **\$set**  
 13467           directive is specified in a message text source file, all messages will be located in an  
 13468           implementation-dependent default message set NL\_SETD (see the **XSH** specification,  
 13469           <nl\_types.h>).

13470       **\$delset** *n* *comment*

13471           This line deletes message set *n* from an existing message catalogue. The *n* denotes the  
 13472           set number [1, {NL\_SETMAX}]. Any string following the set number is treated as a  
 13473           comment.

13474       **\$** *comment*

13475           A line beginning with "\$" followed by a blank character is treated as a comment.

13476       *m* *message-text*

13477           The *m* denotes the message identifier, which is defined as a number in the range [1,  
 13478           {NL\_MSGMAX}] (see the **XSH** specification, <limits.h>). The *message-text* is stored in  
 13479           the message catalogue with the set identifier specified by the last **\$set** directive, and  
 13480           with message identifier *m*. If the *message-text* is empty, and a blank character field  
 13481           separator is present, an empty string is stored in the message catalogue. If a message  
 13482           source line has a message number, but neither a field separator nor *message-text*, the  
 13483           existing message with that number (if any) is deleted from the catalogue. Message  
 13484           identifiers must be in ascending order within a single set, but need not be contiguous.  
 13485           The length of *message-text* must be in the range [0, {NL\_TEXTMAX}] (see the **XSH**  
 13486           specification, <limits.h>).

13487       **\$quote** *n*

13488           This line specifies an optional quote character *c*, which can be used to surround  
 13489           *message-text* so that trailing spaces or null (empty) messages are visible in a message  
 13490           source line. By default, or if an empty **\$quote** directive is supplied, no quoting of  
 13491           *message-text* will be recognised.

13492       Empty lines in a message text source file are ignored. The effects of lines starting with any  
 13493       character other than those defined above are implementation-dependent.

13494 Text strings can contain the special characters and escape sequences defined in the following  
 13495 table:

13496  
 13497

| Description     | Symbol     | Sequence |
|-----------------|------------|----------|
| newline         | NL(LF)     | \n       |
| horizontal tab  | HT         | \t       |
| vertical-tab    | VT         | \v       |
| backspace       | BS         | \b       |
| carriage-return | CR         | \r       |
| form-feed       | FF         | \f       |
| backslash       | \          | \\       |
| bit pattern     | <i>ddd</i> | \ddd     |

13506 The escape sequence \ddd consists of backslash followed by one, two or three octal digits, which  
 13507 are taken to specify the value of the desired character. If the character following a backslash is  
 13508 not one of those specified, the backslash is ignored.

13509 Backslash (\) followed by a newline character is also used to continue a string on the following  
 13510 line. Thus, the following two lines describe a single message string:

13511 1 This line continues \  
 13512 to the next line

13513 which is equivalent to:

13514 1 This line continues to the next line

#### 13515 EXIT STATUS

13516 The following exit values are returned:

13517 0 Successful completion.  
 13518 >0 An error occurred.

#### 13519 CONSEQUENCES OF ERRORS

13520 Default.

#### 13521 APPLICATION USAGE

13522 Message catalogues produced by *gencat* are binary encoded, meaning that their portability  
 13523 cannot be guaranteed between different types of machine. Thus, just as C programs need to be  
 13524 recompiled for each type of machine, so message catalogues must be recreated via *gencat*.

#### 13525 EXAMPLES

13526 None.

#### 13527 FUTURE DIRECTIONS

13528 None.

#### 13529 SEE ALSO

13530 *iconv*, the XSH specification description of <limits.h>.

#### 13531 CHANGE HISTORY

13532 First released in Issue 3.

#### 13533 Issue 4

13534 Format reorganised.

13535 Internationalised environment variable support mandated.

## 13536 NAME

13537 get — get a version of an SCCS file (**DEVELOPMENT**)

## 13538 SYNOPSIS

13539 EX `get [-begkmlPst][-c cutoff][-i list][-r SID][-x list] file...`13540 EX OB `get [-begkmpst][-c cutoff][-i list][ -l[p]][-r SID][-x list] file...`

## 13541 DESCRIPTION

13542 The *get* utility generates a text file from each named SCCS *file* according to the specifications  
13543 given by its options.13544 The generated text is normally written into a file called the *g-file* whose name is derived from the  
13545 SCCS filename by simply removing the leading **s**..

## 13546 OPTIONS

13547 The *get* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
13548 OB that, in the obsolescent form, **-l** option has an optional option-argument that cannot be  
13549 presented as a separate argument (**-lp**). When the **-l** and **-p** options are both needed, the  
13550 application must avoid ambiguity by giving them as separate arguments (**-l -p**), reversing their  
13551 sequence (**-pl**) or separating them with other options in a single argument (such as **-ltb**). The  
13552 following options are supported:13553 **-r *SID*** Indicate the SCCS Identification String (SID) of the version (delta) of an SCCS file to be  
13554 retrieved. The table shows, for the most useful cases, what version of an SCCS file is  
13555 retrieved (as well as the SID of the version to be eventually created by *delta* if the **-e**  
13556 option is also used), as a function of the SID specified.13557 **-c *cutoff***13558 Indicate the *cutoff* date-time, in the form:13559 `YY[MM[DD[HH[MM[SS]]]]]`13560 For the YY component, values in the range [69-99] refer to years in the twentieth  
13561 century (1969 to 1999 inclusive); values in the range [00-68] refer to years in the  
13562 twenty-first century (2000 to 2068 inclusive).13563 No changes (deltas) to the SCCS file that were created after the specified *cutoff* date-  
13564 time are included in the generated text file. Units omitted from the date-time default to  
13565 their maximum possible values; for example, **-c 7502** is equivalent to **-c 750228235959**.13566 Any number of non-numeric characters may separate the various 2-digit pieces of the  
13567 *cutoff* date-time. This feature allows the user to specify a *cutoff* date in the form:  
13568 **-c "77/2/2 9:22:25"**.13569 **-e** Indicate that the *get* is for the purpose of editing or making a change (delta) to the SCCS  
13570 file via a subsequent use of *delta*. The **-e** option used in a *get* for a particular version  
13571 (SID) of the SCCS file prevents further *get* commands from editing on the same SID  
13572 until *delta* is executed or the j (joint edit) flag is set in the SCCS file. Concurrent use of  
13573 *get -e* for different SIDs is always allowed.13574 If the *g-file* generated by *get* with a **-e** option is accidentally ruined in the process of  
13575 editing, it may be regenerated by reexecuting the *get* command with the **-k** option in  
13576 place of the **-e** option.13577 SCCS file protection specified via the ceiling, floor and authorised user list stored in the  
13578 SCCS file is enforced when the **-e** option is used.

13579       **-b**       Use with the **-e** option to indicate that the new delta should have an SID in a new  
 13580 branch as shown in the table below. This option is ignored if the **b** flag is not present in  
 13581 the file or if the retrieved delta is not a leaf delta. (A leaf delta is one that has no  
 13582 successors on the SCCS file tree.)

13583       **Note:**    A branch delta may always be created from a non-leaf delta.

13584       **-i list**   Indicate a *list* of deltas to be included (forced to be applied) in the creation of the  
 13585 generated file. The *list* has the following syntax:

13586               <list> ::= <range> | <list> , <range>  
 13587               <range> ::= SID | SID - SID

13588       SID, the SCCS Identification of a delta, may be in any form shown in the “SID  
 13589 Specified” column of the table below. Partial SIDs are interpreted as shown in the “SID  
 13590 Retrieved” column of the table below.

13591       **-x list**   Indicate a *list* of deltas to be excluded (forced not to be applied) in the creation of the  
 13592 generated file. See the **-i** option for the *list* format.

13593       **-k**       Suppress replacement of identification keywords (see below) in the retrieved text by  
 13594 their value. The **-k** option is implied by the **-e** option.

13595       **-l**       Write a delta summary into an *l-file*.

13596       **-L**       Write a delta summary to standard output. All informative output that normally is  
 13597 written to standard output will be written to standard error instead, unless the **-s**  
 13598 option is used, in which case it is suppressed.

13599 OB       **-lp**     Equivalent to **-L**.

13600       **-p**       Write the text retrieved from the SCCS file to the standard output. No *g-file* is created.  
 13601 All informative output that normally goes to the standard output goes to standard  
 13602 error instead, unless the **-s** option is used, in which case it disappears.

13603       **-s**       Suppress all informative output normally written to standard output. However, fatal  
 13604 error messages (which are always written to the standard error) remain unaffected.

13605       **-m**       Precede each text line retrieved from the SCCS file by the SID of the delta that inserted  
 13606 the text line in the SCCS file. The format is:

13607               "%s\ts", <SID>, <text line>

13608       **-n**       Precede each generated text line with the %M% identification keyword value (see  
 13609 below). The format is:

13610               "%s\ts", <%M% value>, <text line>

13611       When both the **-m** and **-n** options are used, the <text line> is replaced by the **-m**  
 13612 option-generated format.

13613       **-g**       Suppress the actual retrieval of text from the SCCS file. It is primarily used to generate  
 13614 an *l-file*, or to verify the existence of a particular SID.

13615       **-t**       Use to access the most recently created (top) delta in a given release (for example,  
 13616 **-r 1**), or release and level (for example, **-r 1.2**).

## 13617 OPERANDS

13618       The following operands are supported:

13619       **file**     A pathname of an existing SCCS file or a directory. If *file* is a directory, *get* behaves as  
 13620 though each file in the directory were specified as a named file, except that non-SCCS

13621 files (last component of the pathname does not begin with s.) and unreadable files are  
 13622 silently ignored.

13623 If a single instance *file* is specified as *–*, the standard input is read; each line of the  
 13624 standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files  
 13625 and unreadable files are silently ignored.

13626 **STDIN**

13627 The standard input is a text file used only if the *file* operand is specified as *–*. Each line of the  
 13628 text file is interpreted as an SCCS pathname.

13629 **INPUT FILES**

13630 The SCCS files are files of an unspecified format.

13631 **ENVIRONMENT VARIABLES**

13632 The following environment variables affect the execution of *get*:

13633 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 13634 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 13635 default locale will be used. If any of the internationalisation variables contains an  
 13636 invalid setting, the utility will behave as if none of the variables had been defined.

13637 *LC\_ALL*

13638 If set to a non-empty string value, override the values of all the other  
 13639 internationalisation variables.

13640 *LC\_CTYPE*

13641 Determine the locale for the interpretation of sequences of bytes of text data as  
 13642 characters (for example, single- as opposed to multi-byte characters in arguments and  
 13643 input files).

13644 *LC\_MESSAGES*

13645 Determine the locale that should be used to affect the format and contents of diagnostic  
 13646 messages written to standard error, and informative messages written to standard  
 13647 output (or standard error, if the *–p* option is used).

13648 *NLSPATH*

13649 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

13650 **ASYNCHRONOUS EVENTS**

13651 Default.

13652 **STDOUT**

13653 For each file processed, *get* writes to standard output the SID being accessed and the number of  
 13654 lines retrieved from the SCCS file, in the following format:

13655 *"%s\n%d lines\n", <SID>, <number of lines>*

13656 If the *–e* option is used, the SID of the delta to be made appears after the SID accessed and before  
 13657 the number of lines generated, in the POSIX locale:

13658 *"%s\nnew delta %s\n%d\n", <SID accessed>, <SID to be made>,  
 13659 <number of lines>*

13660 If there is more than one named file or if a directory or standard input is named, each pathname  
 13661 is written before each of the lines shown in one of the preceding formats:

13662 *"\n%s:\n", <pathname>*

13663 OB If the *–L* (or *–lp*) option is used, a delta summary will be written following the format specified  
 13664 below for *l-files*.

13665 If the **-i** option is used, included deltas are listed following the notation, in the POSIX locale:

13666 "Included:\n"

13667 If the **-x** option is used, excluded deltas are listed following the notation, in the POSIX locale:

13668 "Excluded:\n"

13669 OB If the **-p**, **-L** or **-lp** options are specified, the standard output consists of the text retrieved from  
13670 the SCCS file.

13671 **STDERR**

13672 The standard error is used only for diagnostic messages, except if the **-p**, **-L** or  
13673 OB **-lp** options are specified, it includes all informative messages normally sent to standard output.

13674 **OUTPUT FILES**

13675 Several auxiliary files may be created by *get*. These files are known generically as the *g-file*, *l-file*,  
13676 *p-file* and *z-file*. The letter before the hyphen is called the *tag*. An auxiliary filename is formed  
13677 from the SCCS file name: the last component of all SCCS filenames must be of the form *s.module-name*;  
13678 the auxiliary files are named by replacing the leading *s* with the tag. The *g-file* is  
13679 an exception to this scheme: the *g-file* is named by removing the *s.* prefix. For example, for  
13680 *s.xyz.c*, the auxiliary filenames would be *xyz.c*, *l.xyz.c*, *p.xyz.c* and *z.xyz.c*, respectively.

13681 The *g-file*, which contains the generated text, is created in the current directory (unless the **-p**  
13682 option is used). A *g-file* is created in all cases, whether or not any lines of text were generated by  
13683 the *get*. It is owned by the real user. If the **-k** option is used or implied, it is writable by the  
13684 owner only (read-only for everyone else); otherwise it is read-only. Only the real user need have  
13685 write permission in the current directory.

13686 The *l-file* contains a table showing which deltas were applied in generating the retrieved text.  
13687 The *l-file* is created in the current directory if the **-l** option is used; it is read-only and it is owned  
13688 by the real user. Only the real user need have write permission in the current directory.

13689 Lines in the *l-file* have the following format:

13690 "%c%c%cΔ%s\t%sΔ%s\n", <code1>, <code2>, <code3>, <SID>, <date-time>,  
13691 <login>

13692 where the entries are:

13693 <code1> A space character if the delta was applied; "\*" otherwise. |

13694 <code2> A space character if the delta was applied or was not applied and ignored; "\*" if the |  
13695 delta was not applied and was not ignored.

13696 <code3> A character indicating a special reason why the delta was or was not applied:

13697 **I** Included.

13698 **X** Excluded.

13699 **C** Cut off (by a **-c** option).

13700 <date-time>

13701 Date and time (using the *date* utility's %y/%m/%d %T format) of creation.

13702 <login> Login name of person who created *delta*.

13703 The comments and MR data follow on subsequent lines, indented one tab character. A blank  
13704 line terminates each entry.



The *p-file* is used to pass information resulting from a *get* with a *-e* option along to *delta*. Its contents are also used to prevent a subsequent execution of *get* with a *-e* option for the same SID until *delta* is executed or the joint edit flag, *j*, is set in the SCCS file. The *p-file* is created in the directory containing the SCCS file and the effective user must have write permission in that directory. It is writable by owner only, and it is owned by the effective user. Each line in the *p-file* has the following format:

```
"%sΔ%sΔ%sΔ%sΔ%sΔ%s\n", <g-file SID>, <SID of new delta>,
<login-name of real user>, <date-time>, <i-value>, <x-value>
```

where *<i-value>* is the value of the *list* option-argument to *-i* (or null) and *<x-value>* is the value of the *list* option-argument to *-x* (or null). There can be an arbitrary number of lines in the *p-file* at any time; no two lines can have the same new delta SID.

The *z-file* serves as a lock-out mechanism against simultaneous updates. Its contents are the binary process ID of the command (that is, *get*) that created it. The *z-file* is created in the directory containing the SCCS file for the duration of *get*. The same protection restrictions as those for the *p-file* apply for the *z-file*. The *z-file* is created read-only.

## EXTENDED DESCRIPTION

| Determination of SCCS Identification String |                       |                                                |                  |                               |
|---------------------------------------------|-----------------------|------------------------------------------------|------------------|-------------------------------|
| SID*<br>Specified                           | -b Keyletter<br>Used† | Other<br>Conditions                            | SID<br>Retrieved | SID of Delta<br>to be Created |
| none‡                                       | no                    | R defaults to mR                               | mR.mL            | mR.(mL+1)                     |
| none‡                                       | yes                   | R defaults to mR                               | mR.mL            | mR.mL.(mB+1).1                |
| R                                           | no                    | R > mR                                         | mR.mL            | R.1***                        |
| R                                           | no                    | R = mR                                         | mR.mL            | mR.(mL+1)                     |
| R                                           | yes                   | R > mR                                         | mR.mL            | mR.mL.(mB+1).1                |
| R                                           | yes                   | R = mR                                         | mR.mL            | mR.mL.(mB+1).1                |
| R                                           | –                     | R < mR and<br>R does not exist                 | hR.mL**          | hR.mL.(mB+1).1                |
| R                                           | –                     | Trunk successor in release > R<br>and R exists | R.mL             | R.mL.(mB+1).1                 |
| R.L                                         | no                    | No trunk successor                             | R.L              | R.(L+1)                       |
| R.L                                         | yes                   | No trunk successor                             | R.L              | R.L.(mB+1).1                  |
| R.L                                         | –                     | Trunk successor<br>in release ≥ R              | R.L              | R.L.(mB+1).1                  |
| R.L.B                                       | no                    | No branch successor                            | R.L.B.mS         | R.L.B.(mS+1)                  |
| R.L.B                                       | yes                   | No branch successor                            | R.L.B.mS         | R.L.(mB+1).1                  |
| R.L.B.S                                     | no                    | No branch successor                            | R.L.B.S          | R.L.B.(S+1)                   |
| R.L.B.S                                     | yes                   | No branch successor                            | R.L.B.S          | R.L.(mB+1).1                  |
| R.L.B.S                                     | –                     | Branch successor                               | R.L.B.S          | R.L.(mB+1).1                  |

\* R, L, B and S are the release, level, branch and sequence components of the SID, respectively; m means maximum. Thus, for example, R.mL means “the maximum level number within release R”; R.L.(mB+1).1 means “the first sequence number on the new branch (that is, maximum branch number plus one) of level L within release R”. Note that if the SID specified is of the form R.L, R.L.B or R.L.B.S, each of the specified components must exist.

13750       \*\*       hR is the highest existing release that is lower than the specified, non-existent, release  
13751       R.

13752       \*\*\*       This is used to force creation of the first delta in a new release.

13753       †        The **-b** option is effective only if the b flag is present in the file. An entry of **-** means  
13754       “irrelevant”.

13755       ‡        This case applies if the d (default SID) flag is not present in the file. If the d flag is  
13756       present in the file, then the SID obtained from the d flag is interpreted as if it had been  
13757       specified on the command line. Thus, one of the other cases in this table applies.

### 13758       **Identification Keywords**

13759       Identifying information is inserted into the text retrieved from the SCCS file by replacing  
13760       identification keywords with their value wherever they occur. The following keywords may be  
13761       used in the text stored in an SCCS file:

13762       **%M %**       Module name: either the value of the m flag in the file, or if absent, the name of the  
13763       SCCS file with the leading s. removed.

13764       **%I %**       SCCS identification (SID) (**%R%.%L%** or **%R%.%L%.%B%.%S%**) of the retrieved  
13765       text

13766       **%R %**       Release.

13767       **%L %**       Level.

13768       **%B %**       Branch.

13769       **%S %**       Sequence.

13770       **%D %**       Current date (**YY/MM/DD**).

13771       **%H %**       Current date (**MM/DD/YY**).

13772       **%T %**       Current time (**HH:MM:SS**).

13773       **%E %**       Date newest applied delta was created (**YY/MM/DD**).

13774       **%G %**       Date newest applied delta was created (**MM/DD/YY**).

13775       **%U %**       Time newest applied delta was created (**HH:MM:SS**).

13776       **%Y %**       Module type: value of the t flag in the SCCS file.

13777       **%F %**       SCCS filename.

13778       **%P %**       SCCS absolute pathname.

13779       **%Q %**       The value of the q flag in the file.

13780       **%C %**       Current line number. This keyword is intended for identifying messages output  
13781       by the program, such as “this should not have happened” type errors. It is not  
13782       intended to be used on every line to provide sequence numbers.

13783       **%Z %**       The four-character string **@(#)** recognisable by *what*.

13784       **%W %**       A shorthand notation for constructing *what* strings:

13785               **%W % = %Z % %M % <tab> %I %**

13786       **%A %**       Another shorthand notation for constructing *what* strings:

13787               **%A % = %Z % %Y % %M % %I % %Z %**

### 13788       **EXIT STATUS**

13789       The following exit values are returned:

13790       0       Successful completion.

13791       >0      An error occurred.

### 13792       **CONSEQUENCES OF ERRORS**

13793       Default.

13794 **APPLICATION USAGE**

13795           None.

13796 **EXAMPLES**

13797           None.

13798 **FUTURE DIRECTIONS**13799           The **-lp** option may be withdrawn in a future issue. |13800 **SEE ALSO**13801           *admin, delta, prs, what.*13802 **CHANGE HISTORY**

13803           First released in Issue 2. |

13804 **Issue 4**

13805           Format reorganised.

13806           Exceptions to Utility Syntax Guidelines conformance noted.

13807           Internationalised environment variable support mandated. |

13808 **Issue 5**

13809           Correction to the first format string in STDOUT. |

13810           The interpretation of the *YY* component of the **-c cutoff** argument is noted.

## 13811 NAME

13812 getconf — get configuration values

## 13813 SYNOPSIS

13814 EX getconf [ -v specification ] *system\_var*13815 EX getconf [ -v specification ] *path\_var pathname*

## 13816 DESCRIPTION

13817 In the first synopsis form, the *getconf* utility will write to the standard output the value of the  
 13818 variable specified by the *system\_var* operand.

13819 In the second synopsis form, the *getconf* utility will write to the standard output the value of the  
 13820 variable specified by the *path\_var* operand for the path specified by the *pathname* operand.

13821 The value of each configuration variable will be determined as if it were obtained by calling the  
 13822 function from which it is defined to be available by this standard or by the **XSH** specification  
 13823 (see the OPERANDS section). The value will reflect conditions in the current operating  
 13824 environment.

## 13825 OPTIONS

13826 EX The *getconf* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 13827 following option is supported:

13828 **-v specification**

13829 Indicate a specific specification and version for which configuration variables are to be  
 13830 determined. If this option is not specified, the values returned will correspond to an  
 13831 implementation default XBS5 conforming compilation environment.

13832 If the command:

13833 getconf \_XBS5\_ILP32\_OFF32

13834 does not write "-1\n" or "undefined\n" to standard output, then commands of the  
 13835 form:

13836 getconf -v XBS5\_ILP32\_OFF32 ...

13837 will determine values for configuration variables corresponding to the  
 13838 XBS5\_ILP32\_OFF32 compilation environment specified in *c89*, EXTENDED  
 13839 DESCRIPTION.

13840 If the command:

13841 getconf \_XBS5\_ILP32\_OFFBIG

13842 does not write "-1\n" or "undefined\n" to standard output, then commands of the  
 13843 form:

13844 getconf -v XBS5\_ILP32\_OFFBIG ...

13845 will determine values for configuration variables corresponding to the  
 13846 XBS5\_ILP32\_OFFBIG compilation environment specified in *c89*, EXTENDED  
 13847 DESCRIPTION.

13848 If the command:

13849 getconf \_XBS5\_LP64\_OFF64

13850 does not write "-1\n" or "undefined\n" to standard output, then commands of the  
 13851 form:

|          |                                                                                                              |  |
|----------|--------------------------------------------------------------------------------------------------------------|--|
| 13852    | <code>getconf -v XBS5_LP64_OFF64 ...</code>                                                                  |  |
| 13853    | will determine values for configuration variables corresponding to the                                       |  |
| 13854    | XBS5_LP64_OFF64 compilation environment specified in <i>c89</i> , EXTENDED                                   |  |
| 13855    | DESCRIPTION.                                                                                                 |  |
| 13856    | If the command:                                                                                              |  |
| 13857    | <code>getconf _XBS5_LPBIG_OFFBIG</code>                                                                      |  |
| 13858    | does not write "-1\n" or "undefined\n" to standard output, then commands of the                              |  |
| 13859    | form:                                                                                                        |  |
| 13860    | <code>getconf -v _XBS5_LPBIG_OFFBIG</code>                                                                   |  |
| 13861    | will determine values for configuration variables corresponding to the                                       |  |
| 13862    | XBS5_LPBIG_OFFBIG compilation environment specified in <i>c89</i> , EXTENDED                                 |  |
| 13863    | DESCRIPTION.                                                                                                 |  |
| 13864    |                                                                                                              |  |
| 13865    | <b>OPERANDS</b>                                                                                              |  |
| 13866    | The following operands are supported:                                                                        |  |
| 13867    | <i>path_var</i>                                                                                              |  |
| 13868    | A name of a configuration variable. All of the variables in the <b>XSH</b> specification,                    |  |
| 13869    | <i>pathconf()</i> , DESCRIPTION are supported and the implementation may add other local                     |  |
| 13870    | variables.                                                                                                   |  |
| 13871    | <i>pathname</i>                                                                                              |  |
| 13872    | A pathname for which the variable specified by <i>path_var</i> is to be determined.                          |  |
| 13873    | <i>system_var</i>                                                                                            |  |
| 13874    | A name of a configuration variable. All of the variables in the <b>XSH</b> specification,                    |  |
| 13875    | <i>confstr()</i> and <i>sysconf()</i> , DESCRIPTIONs are supported and the implementation may                |  |
| 13876    | add other local values:                                                                                      |  |
| 13877 EX | When the symbol listed in the first column of the following table is used as the                             |  |
| 13878    | <i>system_var</i> operand, <i>getconf</i> will yield the same value as <i>confstr()</i> when called with the |  |
| 13879    | value in the second column:                                                                                  |  |

|       |                             |                                 |
|-------|-----------------------------|---------------------------------|
| 13880 |                             |                                 |
| 13881 |                             |                                 |
| 13882 |                             |                                 |
| 13883 |                             |                                 |
| 13884 |                             |                                 |
| 13885 |                             |                                 |
| 13886 |                             |                                 |
| 13887 |                             |                                 |
| 13888 |                             |                                 |
| 13889 |                             |                                 |
| 13890 |                             |                                 |
| 13891 |                             |                                 |
| 13892 |                             |                                 |
| 13893 |                             |                                 |
| 13894 |                             |                                 |
| 13895 |                             |                                 |
| 13896 |                             |                                 |
| 13897 |                             |                                 |
| 13898 |                             |                                 |
|       | <i>system_var</i>           | <i>confstr() Name Value</i>     |
|       | PATH                        | _CS_PATH                        |
|       | XBS5_ILP32_OFF32_CFLAGS     | _CS_XBS5_ILP32_OFF32_CFLAGS     |
|       | XBS5_ILP32_OFF32_LDFLAGS    | _CS_XBS5_ILP32_OFF32_LDFLAGS    |
|       | XBS5_ILP32_OFF32_LIBS       | _CS_XBS5_ILP32_OFF32_LIBS       |
|       | XBS5_ILP32_OFF32_LINTFLAGS  | _CS_XBS5_ILP32_OFF32_LINTFLAGS  |
|       | XBS5_ILP32_OFFBIG_CFLAGS    | _CS_XBS5_ILP32_OFFBIG_CFLAGS    |
|       | XBS5_ILP32_OFFBIG_LDFLAGS   | _CS_XBS5_ILP32_OFFBIG_LDFLAGS   |
|       | XBS5_ILP32_OFFBIG_LIBS      | _CS_XBS5_ILP32_OFFBIG_LIBS      |
|       | XBS5_ILP32_OFFBIG_LINTFLAGS | _CS_XBS5_ILP32_OFFBIG_LINTFLAGS |
|       | XBS5_LP64_OFF64_CFLAGS      | _CS_XBS5_LP64_OFF64_CFLAGS      |
|       | XBS5_LP64_OFF64_LDFLAGS     | _CS_XBS5_LP64_OFF64_LDFLAGS     |
|       | XBS5_LP64_OFF64_LIBS        | _CS_XBS5_LP64_OFF64_LIBS        |
|       | XBS5_LP64_OFF64_LINTFLAGS   | _CS_XBS5_LP64_OFF64_LINTFLAGS   |
|       | XBS5_LPBIG_OFFBIG_CFLAGS    | _CS_XBS5_LPBIG_OFFBIG_CFLAGS    |
|       | XBS5_LPBIG_OFFBIG_LDFLAGS   | _CS_XBS5_LPBIG_OFFBIG_LDFLAGS   |
|       | XBS5_LPBIG_OFFBIG_LIBS      | _CS_XBS5_LPBIG_OFFBIG_LIBS      |
|       | XBS5_LPBIG_OFFBIG_LINTFLAGS | _CS_XBS5_LPBIG_OFFBIG_LINTFLAGS |

13899 **STDIN**

13900 Not used.

13901 **INPUT FILES**

13902 None.

13903 **ENVIRONMENT VARIABLES**13904 The following environment variables affect the execution of *getconf*:

13905 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 13906 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 13907 default locale will be used. If any of the internationalisation variables contains an  
 13908 invalid setting, the utility will behave as if none of the variables had been defined.

13909 *LC\_ALL*

13910 If set to a non-empty string value, override the values of all the other  
 13911 internationalisation variables.

13912 *LC\_CTYPE*

13913 Determine the locale for the interpretation of sequences of bytes of text data as  
 13914 characters (for example, single- as opposed to multi-byte characters in arguments).

13915 *LC\_MESSAGES*

13916 Determine the locale that should be used to affect the format and contents of diagnostic  
 13917 messages written to standard error.

13918 EX *NLSPATH*13919 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.13920 **ASYNCHRONOUS EVENTS**

13921 Default.

13922 **STDOUT**

13923 If the specified variable is defined on the system and its value is described to be available from  
 13924 the **XSH** specification *confstr()* function, its value will be written in the following format:

13925 "%s\n", &lt;value&gt;

13926 Otherwise, if the specified variable is defined on the system, its value will be written in the  
 13927 following format:

13928       "%d\n", <value>

13929 If the specified variable is valid, but is undefined on the system, *getconf* will write using the  
 13930 following format:

13931       "undefined\n"

13932 If the variable name is invalid or an error occurs, nothing will be written to standard output.

#### 13933 **STDERR**

13934       Used only for diagnostic messages.

#### 13935 **OUTPUT FILES**

13936       None.

#### 13937 **EXTENDED DESCRIPTION**

13938       None.

#### 13939 **EXIT STATUS**

13940       The following exit values are returned:

13941       0 The specified variable is valid and information about its current state was written  
 13942       successfully.

13943       >0 An error occurred.

#### 13944 **CONSEQUENCES OF ERRORS**

13945       Default.

#### 13946 **APPLICATION USAGE**

13947       None.

#### 13948 **EXAMPLES**

13949       This example illustrates the value of {NGROUPS\_MAX}:

13950       getconf NGROUPS\_MAX

13951       This example illustrates the value of {NAME\_MAX} for a specific directory:

13952       getconf NAME\_MAX /usr

13953       This example shows how to deal more carefully with results that might be unspecified:

```
13954 if value=$(getconf PATH_MAX /usr); then
13955 if ["$value" = "undefined"]; then
13956 echo PATH_MAX in /usr is infinite.
13957 else
13958 echo PATH_MAX in /usr is $value.
13959 fi
13960 else
13961 echo Error in getconf.
13962 fi
```

13963       Note that:

13964       sysconf(\_SC\_POSIX\_C\_BIND);

13965 and:

13966 `system("getconf POSIX2_C_BIND");`

13967 in a C program could give different answers. The *sysconf()* call supplies a value that  
 13968 corresponds to the conditions when the program was either compiled or executed, depending on  
 13969 the implementation; the *system()* call to *getconf* always supplies a value corresponding to  
 13970 conditions when the program is executed.

#### 13971 **FUTURE DIRECTIONS**

13972 None.

#### 13973 **SEE ALSO**

13974 *c89* and the **XSH** specification description of *confstr()*, *pathconf()*, *sysconf()*.

#### 13975 **CHANGE HISTORY**

13976 First released in Issue 4.

#### 13977 **Issue 4, Version 2**

13978 The following changes are made in the table of values for *system\_var*:

- 13979 • Names beginning with `POSIX_` are changed to begin with `_POSIX_`.
- 13980 • Names beginning with `XOPEN_` are changed to begin with `_XOPEN_`.
- 13981 • `MN_NMAX` is changed to `NL_MAX`.
- 13982 • `NL_SET_MAX` is changed to `NL_SETMAX`.
- 13983 • `NL_TEXT_MAX` is changed to `NL_TEXTMAX`.
- 13984 • The `_XOPEN_CRYPT`, `_XOPEN_ENH_I18N` and `_XOPEN_SHM` configuration variables are  
 13985 added to the list.

#### 13986 **Issue 5**

13987 In the OPERANDS section:

- 13988 • `NL_MAX` is changed to `NL_NMAX`.
- 13989 • Entries beginning `NL_` are deleted from the list of standard configuration variables.
- 13990 • The list of variables previously marked `UX` is merged with the list marked `EX`.
- 13991 • Operands are added to support new Feature Groups.
- 13992 • Operands are added so that *getconf* can determine supported programming environments.



13993 **NAME**

13994       getopts — parse utility options

13995 **SYNOPSIS**13996       getopts *optstring name* [*arg...*]13997 **DESCRIPTION**

13998       The *getopts* utility can be used to retrieve options and option-arguments from a list of  
 13999       parameters. It supports the utility argument syntax guidelines 3 to 10, inclusive, described in  
 14000       the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

14001       Each time it is invoked, the *getopts* utility places the value of the next option in the shell variable  
 14002       specified by the *name* operand and the index of the next argument to be processed in the shell  
 14003       variable *OPTIND*. Whenever the shell is invoked, *OPTIND* will be initialised to 1.

14004       When the option requires an option-argument, the *getopts* utility will place it in the shell variable  
 14005       *OPTARG*. If no option was found, or if the option that was found does not have an option-  
 14006       argument, *OPTARG* will be unset.

14007       If an option character not contained in the *optstring* operand is found where an option character  
 14008       is expected, the shell variable specified by *name* will be set to the question-mark (?) character. In  
 14009       this case, if the first character in *optstring* is a colon (:), the shell variable *OPTARG* will be set to  
 14010       the option character found, but no output will be written to standard error; otherwise, the shell  
 14011       variable *OPTARG* will be unset and a diagnostic message will be written to standard error. This  
 14012       condition is considered to be an error detected in the way arguments were presented to the  
 14013       invoking application, but is not an error in *getopts* processing.

14014       If an option-argument is missing:

- 14015       • If the first character of *optstring* is a colon, the shell variable specified by *name* will be set to  
 14016       the colon character and the shell variable *OPTARG* will be set to the option character found.
- 14017       • Otherwise, the shell variable specified by *name* will be set to the question-mark character, the  
 14018       shell variable *OPTARG* will be unset, and a diagnostic message will be written to standard  
 14019       error. This condition is considered to be an error detected in the way arguments were  
 14020       presented to the invoking application, but is not an error in *getopts* processing; a diagnostic  
 14021       message will be written as stated, but the exit status will be zero.

14022       When the end of options is encountered, the *getopts* utility will exit with a return value greater  
 14023       than zero; the shell variable *OPTIND* will be set to the index of the first non-option-argument,  
 14024       where the first -- argument is considered to be an option-argument if there are no other non-  
 14025       option-arguments appearing before it, or the value \$# + 1 if there are no non-option-arguments;  
 14026       the *name* variable will be set to the question-mark character. Any of the following identifies the  
 14027       end of options: the special option --, finding an argument that does not begin with a "-", or  
 14028       encountering an error.

14029       The shell variables *OPTIND* and *OPTARG* are local to the caller of *getopts* and are not exported  
 14030       by default.

14031       The shell variable specified by the *name* operand, *OPTIND* and *OPTARG* affect the current shell  
 14032       execution environment; see Section 2.12 on page 63.

14033       If the application sets *OPTIND* to the value 1, a new set of parameters can be used: either the  
 14034       current positional parameters or new *arg* values. Any other attempt to invoke *getopts* multiple  
 14035       times in a single shell execution environment with parameters (positional parameters or *arg*  
 14036       operands) that are not the same in all invocations, or with an *OPTIND* value modified to be a  
 14037       value other than 1, produces unspecified results.

14038 **OPTIONS**

14039       None.

14040 **OPERANDS**

14041       The following operands are supported:

14042       *optstring*

14043       A string containing the option characters recognised by the utility invoking *getopts*. If a  
 14044       character is followed by a colon, the option will be expected to have an argument,  
 14045       which should be supplied as a separate argument. Applications should specify an  
 14046       option character and its option-argument as separate arguments, but *getopts* will  
 14047       interpret the characters following an option character requiring arguments as an  
 14048       argument whether or not this is done. An explicit null option-argument need not be  
 14049       recognised if it is not supplied as a separate argument when *getopts* is invoked. (See  
 14050       also the **XSH** specification *getopt()* function.) The characters question-mark and colon  
 14051       must not be used as option characters by an application. The use of other option  
 14052       characters that are not alphanumeric produces unspecified results. If the option-  
 14053       argument is not supplied as a separate argument from the option character, the value  
 14054       in *OPTARG* will be stripped of the option character and the "-". The first character in  
 14055       *optstring* will determine how *getopts* will behave if an option character is not known or  
 14056       an option-argument is missing.

14057       *name*     The name of a shell variable that will be set by the *getopts* utility to the option character  
 14058       that was found.

14059       The *getopts* utility by default will parse positional parameters passed to the invoking shell  
 14060       procedure. If *args* are given, they will be parsed instead of the positional parameters.

14061 **STDIN**

14062       Not used.

14063 **INPUT FILES**

14064       None.

14065 **ENVIRONMENT VARIABLES**14066       The following environment variables affect the execution of *getopts*:

14067       *LANG*     Provide a default value for the internationalisation variables that are unset or null. If  
 14068       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 14069       default locale will be used. If any of the internationalisation variables contains an  
 14070       invalid setting, the utility will behave as if none of the variables had been defined.

14071       *LC\_ALL*

14072       If set to a non-empty string value, override the values of all the other  
 14073       internationalisation variables.

14074       *LC\_CTYPE*

14075       Determine the locale for the interpretation of sequences of bytes of text data as  
 14076       characters (for example, single- as opposed to multi-byte characters in arguments and  
 14077       input files).

14078       *LC\_MESSAGES*

14079       Determine the locale that should be used to affect the format and contents of diagnostic  
 14080       messages written to standard error.

14081 EX     *NLSPATH*14082       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14083           **OPTIND**  
 14084           This variable will be used by the *getopts* utility as the index of the next argument to be  
 14085           processed.

14086 **ASYNCHRONOUS EVENTS**  
 14087           Default.

14088 **STDOUT**  
 14089           Not used.

14090 **STDERR**  
 14091           Whenever an error is detected and the first character in the *optstring* operand is not a colon (:), a  
 14092           diagnostic message will be written to standard error with the following information in an  
 14093           unspecified format:

- 14094           • The invoking program name will be identified in the message. The invoking program name  
 14095           will be the value of the shell special parameter 0 (see Section 2.5.2 on page 27) at the time the  
 14096           *getopts* utility is invoked. A name equivalent to:  
 14097           baseline "\$0"  
 14098           may be used.
- 14099           • If an option is found that was not specified in *optstring*, this error will be identified and the  
 14100           invalid option character will be identified in the message.
- 14101           • If an option requiring an option-argument is found, but an option-argument is not found,  
 14102           this error will be identified and the invalid option character will be identified in the message.

14103 **OUTPUT FILES**  
 14104           None.

14105 **EXTENDED DESCRIPTION**  
 14106           None.

14107 **EXIT STATUS**  
 14108           The following exit values are returned:

- 14109           0   An option, specified or unspecified by *optstring*, was found.
- 14110           >0 The end of options was encountered or an error occurred.

14111 **CONSEQUENCES OF ERRORS**  
 14112           Default.

14113 **APPLICATION USAGE**  
 14114           Since *getopts* affects the current shell execution environment, it is generally provided as a shell  
 14115           regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
 14116           of the following:

```
14117 (getopts abc value "$@")
14118 nohup getopts ...
14119 find . -exec getopts ... \;
```

14120           it will not affect the shell variables in the caller's environment.

14121           Note that shell functions share *OPTIND* with the calling shell even though the positional  
 14122           parameters are changed. Functions that want to use *getopts* to parse their arguments will usually  
 14123           want to save the value of *OPTIND* on entry and restore it before returning. However, there will  
 14124           be cases when a function will want to change *OPTIND* for the calling shell.

14125 **EXAMPLES**

14126       The following example script parses and displays its arguments:

```

14127 aflag=
14128 bflag=
14129 while getopts ab: name
14130 do
14131 case $name in
14132 a) aflag=1;;
14133 b) bflag=1
14134 bval="$OPTARG";;
14135 ?) printf "Usage: %s: [-a] [-b value] args\n" $0
14136 exit 2;;
14137 esac
14138 done
14139 if [! -z "$aflag"]; then
14140 printf "Option -a specified\n"
14141 fi
14142 if [! -z "$bflag"]; then
14143 printf 'Option -b "%s" specified\n' "$bval"
14144 fi
14145 shift $(($OPTIND - 1))
14146 printf "Remaining arguments are: %s\n" "$*"

```

14147 **FUTURE DIRECTIONS**

14148       None.

14149 **SEE ALSO**

14150       The **XSH** specification description of *getopt()*.

14151 **CHANGE HISTORY**

14152       First released in Issue 4.

## 14153 NAME

14154 `grep` — search a file for a pattern

## 14155 SYNOPSIS

```
14156 grep [-E | -F][-c | -l | -q][-insvx] -e pattern_list
14157 [-f pattern_file]...[file...]
```

```
14158 grep [-E | -F][-c | -l | -q][-insvx][-e pattern_list
14159 -f pattern_file]...[file...]
```

```
14160 grep [-E | -F][-c | -l | -q][-insvx] pattern_list[file...]
```

## 14161 DESCRIPTION

14162 The *grep* utility searches the input files, selecting lines matching one or more patterns; the types  
 14163 of patterns are controlled by the options specified. The patterns are specified by the `-e` option,  
 14164 `-f` option, or the *pattern\_list* operand. The *pattern\_list*'s value consists of one or more patterns  
 14165 separated by newline characters; the *pattern\_file*'s contents consist of one or more patterns  
 14166 terminated by newline characters. By default, an input line will be selected if any pattern,  
 14167 treated as an entire basic regular expression (BRE) as described in the **XBD** specification, **Section**  
 14168 **7.3, Basic Regular Expressions**, matches any part of the line; a null BRE will match every line.  
 14169 By default, each selected input line will be written to the standard output.

14170 Regular expression matching will be based on text lines. Since a newline character separates or  
 14171 terminates patterns (see the `-e` and `-f` options below), regular expressions cannot contain a  
 14172 newline character. Similarly, since patterns are matched against individual lines of the input,  
 14173 there is no way for a pattern to match a newline character found in the input.

## 14174 OPTIONS

14175 The *grep* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

14176 The following options are supported:

14177 **-E** Match using extended regular expressions. Treat each pattern specified as an ERE, as  
 14178 described in the **XBD** specification, **Section 7.4, Extended Regular Expressions**. If any  
 14179 entire ERE pattern matches an input line, the line will be matched. A null ERE matches  
 14180 every line.

14181 **-F** Match using fixed strings. Treat each pattern specified as a string instead of a regular  
 14182 expression. If an input line contains any of the patterns as a contiguous sequence of  
 14183 bytes, the line will be matched. A null string matches every line.

14184 **-c** Write only a count of selected lines to standard output.

14185 **-e *pattern\_list***

14186 Specify one or more patterns to be used during the search for input. Patterns in  
 14187 *pattern\_list* must be separated by a newline character. A null pattern can be specified  
 14188 by two adjacent newline characters in *pattern\_list*. Unless the `-E` or `-F` option is also  
 14189 specified, each pattern will be treated as a BRE, as described in the **XBD** specification,  
 14190 **Section 7.3, Basic Regular Expressions**. Multiple `-e` and `-f` options are accepted by the  
 14191 *grep* utility. All of the specified patterns are used when matching lines, but the order of  
 14192 evaluation is unspecified.

14193 **-f *pattern\_file***

14194 Read one or more patterns from the file named by the pathname *pattern\_file*. Patterns  
 14195 in *pattern\_file* are terminated by a newline character. A null pattern can be specified by  
 14196 an empty line in *pattern\_file*. Unless the `-E` or `-F` option is also specified, each pattern  
 14197 will be treated as a BRE, as described in the **XBD** specification, **Section 7.3, Basic**  
 14198 **Regular Expressions**.

- 14199        **-i**        Perform pattern matching in searches without regard to case. See the **XBD**  
 14200                   specification, **Section 7.2, Regular Expression General Requirements**.
- 14201        **-l**        (The letter ell.) Write only the names of files containing selected lines to standard  
 14202                   output. Pathnames are written once per file searched. If the standard input is searched,  
 14203                   a pathname of "**(standard input)**" will be written, in the POSIX locale. In other locales,  
 14204                   **standard input** may be replaced by something more appropriate in those locales.
- 14205        **-n**        Precede each output line by its relative line number in the file, each file starting at line  
 14206                   1. The line number counter will be reset for each file processed.
- 14207        **-q**        Quiet. Do not write anything to the standard output, regardless of matching lines. Exit  
 14208                   with zero status if an input line is selected.
- 14209        **-s**        Suppress the error messages ordinarily written for non-existent or unreadable files.  
 14210                   Other error messages will not be suppressed.
- 14211        **-v**        Select lines not matching any of the specified patterns. If the **-v** option is not specified,  
 14212                   selected lines will be those that match any of the specified patterns.
- 14213        **-x**        Consider only input lines that use all characters in the line to match an entire fixed  
 14214                   string or regular expression to be matching lines.

#### 14215 OPERANDS

14216        The following operands are supported:

- 14217        *pattern*    Specify one or more patterns to be used during the search for input. This operand is  
 14218                   treated as if it were specified as **-e *pattern\_list***.
- 14219        *file*        A pathname of a file to be searched for the patterns. If no *file* operands are specified,  
 14220                   the standard input will be used.

#### 14221 STDIN

14222        The standard input will be used only if no *file* operands are specified. See the INPUT FILES  
 14223                   section.

#### 14224 INPUT FILES

14225        The input files must be text files.

#### 14226 ENVIRONMENT VARIABLES

14227        The following environment variables affect the execution of *grep*:

- 14228        *LANG*        Provide a default value for the internationalisation variables that are unset or null. If  
 14229                   *LANG* is unset or null, the corresponding value from the implementation-dependent  
 14230                   default locale will be used. If any of the internationalisation variables contains an  
 14231                   invalid setting, the utility will behave as if none of the variables had been defined.

#### 14232 LC\_ALL

14233        If set to a non-empty string value, override the values of all the other  
 14234                   internationalisation variables.

#### 14235 LC\_COLLATE

14236        Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 14237                   character collating elements within regular expressions.

#### 14238 LC\_CTYPE

14239        Determine the locale for the interpretation of sequences of bytes of text data as  
 14240                   characters (for example, single- as opposed to multi-byte characters in arguments and  
 14241                   input files) and the behaviour of character classes within regular expressions.

14242 **LC\_MESSAGES**

14243 Determine the locale that should be used to affect the format and contents of diagnostic  
 14244 messages written to standard error.

14245 EX **NLSPATH**

14246 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14247 **ASYNCHRONOUS EVENTS**

14248 Default.

14249 **STDOUT**

14250 If the **-l** option is in effect, and the **-q** option is not, the following will be written for each file  
 14251 containing at least one selected input line:

14252 `"%s\n", file`

14253 Otherwise, if more than one *file* argument appears, and **-q** is not specified, the *grep* utility will  
 14254 prefix each output line by:

14255 `"%s: ", file`

14256 The remainder of each output line depends on the other options specified:

- 14257 • If the **-c** option is in effect, the remainder of each output line will contain:

14258 `"%d\n", <count>`

- 14259 • Otherwise, if **-c** is not in effect and the **-n** option is in effect, the following will be written to  
 14260 standard output:

14261 `"%d: ", <line number>`

- 14262 • Finally, the following will be written to standard output:

14263 `"%s", <selected-line contents>`

14264 **STDERR**

14265 Used only for diagnostic messages.

14266 **OUTPUT FILES**

14267 None.

14268 **EXTENDED DESCRIPTION**

14269 None.

14270 **EXIT STATUS**

14271 The following exit values are returned:

14272 0 One or more lines were selected.

14273 1 No lines were selected.

14274 >1 An error occurred.

14275 **CONSEQUENCES OF ERRORS**

14276 If the **-q** option is specified, the exit status will be zero if an input line is selected, even if an error  
 14277 was detected. Otherwise, default actions will be performed.

14278 **APPLICATION USAGE**

14279 Care should be taken when using characters in *pattern\_list* that may also be meaningful to the  
 14280 command interpreter. It is safest to enclose the entire *pattern\_list* argument in single quotes:

14281 `'...'`

14282 The **-e** *pattern\_list* option has the same effect as the *pattern\_list* operand, but is useful when  
 14283 *pattern\_list* begins with the hyphen delimiter. It is also useful when it is more convenient to  
 14284 provide multiple patterns as separate arguments.

14285 Multiple **-e** and **-f** options are accepted and *grep* will use all of the patterns it is given while  
 14286 matching input text lines. (Note that the order of evaluation is not specified. If an  
 14287 implementation finds a null string as a pattern, it is allowed to use that pattern first, matching  
 14288 every line, and effectively ignore any other patterns.)

14289 The **-q** option provides a means of easily determining whether or not a pattern (or string) exists  
 14290 in a group of files. When searching several files, it provides a performance improvement  
 14291 (because it can quit as soon as it finds the first match) and requires less care by the user in  
 14292 choosing the set of files to supply as arguments (because it will exit zero if it finds a match even  
 14293 if *grep* detected an access or read error on earlier file operands).

#### 14294 EXAMPLES

14295 1. To find all uses of the word Posix (in any case) in file **text.mm** and write with line numbers:

```
14296 grep -i -n posix text.mm
```

14297 2. To find all empty lines in the standard input:

```
14298 grep ^$
```

14299 or:

```
14300 grep -v .
```

14301 3. Both of the following commands print all lines containing strings **abc** or **def** or both:

```
14302 grep -E 'abc
14303 def'
```

```
14304 grep -F 'abc
14305 def'
```

14306 4. Both of the following commands print all lines matching exactly **abc** or **def**:

```
14307 grep -E '^abc$
14308 ^def$'
```

```
14309 grep -F -x 'abc
14310 def'
```

#### 14311 FUTURE DIRECTIONS

14312 None.

#### 14313 SEE ALSO

14314 *sed*.

#### 14315 CHANGE HISTORY

14316 First released in Issue 2.

#### 14317 Issue 4

14318 Aligned with the ISO/IEC 9945-2: 1993 standard.



14319 **NAME**

14320 hash — remember or report utility locations

14321 **SYNOPSIS**14322 EX hash [*utility...*]

14323 EX hash -r

14324 **DESCRIPTION**

14325 The *hash* utility affects the way the current shell environment remembers the locations of utilities  
 14326 found as described in **Command Search and Execution** on page 47. Depending on the  
 14327 arguments specified, it adds utility locations to its list of remembered locations or it purges the  
 14328 contents of the list. When no arguments are specified, it reports on the contents of the list.

14329 Utilities provided as built-ins to the shell are not reported by *hash*.14330 **OPTIONS**14331 The *hash* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

14332 The following option is supported:

14333 -r Forget all previously remembered utility locations.

14334 **OPERANDS**

14335 The following operand is supported:

14336 *utility* The name of a utility to be searched for and added to the list of remembered locations.  
 14337 If *utility* contains one or more slashes, the results are unspecified.

14338 **STDIN**

14339 Not used.

14340 **INPUT FILES**

14341 None.

14342 **ENVIRONMENT VARIABLES**14343 The following environment variables affect the execution of *hash*:

14344 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 14345 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 14346 default locale will be used. If any of the internationalisation variables contains an  
 14347 invalid setting, the utility will behave as if none of the variables had been defined.

14348 *LC\_ALL*

14349 If set to a non-empty string value, override the values of all the other  
 14350 internationalisation variables.

14351 *LC\_CTYPE*

14352 Determine the locale for the interpretation of sequences of bytes of text data as  
 14353 characters (for example, single- as opposed to multi-byte characters in arguments).

14354 *LC\_MESSAGES*

14355 Determine the locale that should be used to affect the format and contents of diagnostic  
 14356 messages written to standard error.

14357 *NLSPATH*14358 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14359 *PATH* Determine the location of *utility*, as described in the **XBD** specification, **Chapter 6,**  
 14360 **Environment Variables**.

14361 **ASYNCHRONOUS EVENTS**

14362 Default.

14363 **STDOUT**

14364 The standard output of *hash* is used when no arguments are specified. Its format is unspecified,  
14365 but includes the pathname of each utility in the list of remembered locations for the current shell  
14366 environment. This list consists of those utilities named in previous *hash* invocations that have  
14367 been invoked, and may contain those invoked and found through the normal command search  
14368 process.

14369 **STDERR**

14370 Used only for diagnostic messages.

14371 **OUTPUT FILES**

14372 None.

14373 **EXTENDED DESCRIPTION**

14374 None.

14375 **EXIT STATUS**

14376 The following exit values are returned:

14377 0 Successful completion.

14378 &gt;0 An error occurred.

14379 **CONSEQUENCES OF ERRORS**

14380 Default.

14381 **APPLICATION USAGE**

14382 Since *hash* affects the current shell execution environment, it is always provided as a shell  
14383 regular built-in. If it is called in a separate utility execution environment, such as one of the  
14384 following:

14385 nohup hash -r

14386 find . -type f | xargs hash

14387 it will not affect the command search process of the caller's environment.

14388 The *hash* utility may be implemented as an alias, for example, *alias -t -*, in which case utilities  
14389 found through normal command search will not be listed by the *hash* command.

14390 The effects of *hash -r* can also be achieved portably by resetting the value of *PATH*; in the  
14391 simplest form, this can be:

14392 PATH="\$PATH"

14393 The use of *hash* with *utility* names is unnecessary for most applications, but may provide a  
14394 performance improvement on a few implementations; normally, the hashing process is included  
14395 by default.

14396 **EXAMPLES**

14397 None.

14398 **FUTURE DIRECTIONS**

14399 None.

14400 **SEE ALSO**14401 **Command Search and Execution** on page 47.

14402 **CHANGE HISTORY**

14403 First released in Issue 2.

|

14404 **Issue 4**14405 Relocated from the *sh* description to reflect its status as a regular built-in utility.

## 14406 NAME

14407 head — copy the first part of files

## 14408 SYNOPSIS

14409 head [-n *number*][*file...*]

14410 OB head [*number*][*file...*]

## 14411 DESCRIPTION

14412 The *head* utility will copy its input files to the standard output, ending the output for each file at  
14413 a designated point.

14414 OB Copying will end at the point in each input file indicated by the **-n *number*** option (or the  
14415 **-*number*** argument). The option-argument *number* will be counted in units of lines.

## 14416 OPTIONS

14417 OB The *head* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
14418 that the obsolescent version accepts multi-character numeric options.

14419 The following options are supported:

14420 **-n *number***

14421 The first *number* lines of each input file will be copied to standard output. The *number*  
14422 option-argument must be a positive decimal integer.

14423 OB **-*number***

14424 The *number* argument is a positive decimal integer with the same effect as the  
14425 **-n *number*** option.

14426 If no options are specified, *head* will act as if **-n 10** had been specified.

## 14427 OPERANDS

14428 The following operand is supported:

14429 *file* A pathname of an input file. If no *file* operands are specified, the standard input will be  
14430 used.

## 14431 STDIN

14432 The standard input will be used only if no *file* operands are specified. See the INPUT FILES  
14433 section.

## 14434 INPUT FILES

14435 Input files must be text files, but the line length is not restricted to {LINE\_MAX} bytes.

## 14436 ENVIRONMENT VARIABLES

14437 The following environment variables affect the execution of *head*:

14438 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
14439 *LANG* is unset or null, the corresponding value from the implementation-dependent  
14440 default locale will be used. If any of the internationalisation variables contains an  
14441 invalid setting, the utility will behave as if none of the variables had been defined.

14442 *LC\_ALL*

14443 If set to a non-empty string value, override the values of all the other  
14444 internationalisation variables.

14445 *LC\_CTYPE*

14446 Determine the locale for the interpretation of sequences of bytes of text data as  
14447 characters (for example, single- as opposed to multi-byte characters in arguments and  
14448 input files).

14449 **LC\_MESSAGES**  
 14450 Determine the locale that should be used to affect the format and contents of diagnostic  
 14451 messages written to standard error.

14452 EX **NLSPATH**  
 14453 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14454 **ASYNCHRONOUS EVENTS**  
 14455 Default.

14456 **STDOUT**  
 14457 The standard output will contain designated portions of the input files.  
 14458 If multiple *file* operands are specified, *head* will precede the output for each with the header:  
 14459 `"\n==> %s <==\n", <pathname>`  
 14460 except that the first header written will not include the initial newline character.

14461 **STDERR**  
 14462 Used only for diagnostic messages.

14463 **OUTPUT FILES**  
 14464 None.

14465 **EXTENDED DESCRIPTION**  
 14466 None.

14467 **EXIT STATUS**  
 14468 The following exit values are returned:  
 14469 0 Successful completion.  
 14470 >0 An error occurred.

14471 **CONSEQUENCES OF ERRORS**  
 14472 Default.

14473 **APPLICATION USAGE**  
 14474 None.

14475 **EXAMPLES**  
 14476 To write the first ten lines of all files (except those with a leading period) in the directory:  
 14477 `head *`

14478 **FUTURE DIRECTIONS**  
 14479 The obsolescent *–number* form may be withdrawn in a future issue. Applications should use the  
 14480 *–n number* option.

14481 **SEE ALSO**  
 14482 *sed*, *tail*.

14483 **CHANGE HISTORY**  
 14484 First released in Issue 4.

14485 **NAME**

14486 iconv — codeset conversion

14487 **SYNOPSIS**14488 EX `iconv -f fromcode -t tocode [file...]`14489 **DESCRIPTION**14490 The *iconv* utility converts the encoding of characters in *file* from one codeset to another and  
14491 writes the results to standard output.14492 Character encodings in either codeset may include single-byte values (for example, for the ISO  
14493 8859-1: 1987 standard characters) or multi-byte values (for example, for certain characters in the  
14494 ISO 6937: 1983 standard). The results of specifying invalid characters in the input stream (either  
14495 those that are not valid members of the *fromcode* or those that have no corresponding value in  
14496 *tocode*) are specified in the system documentation.14497 **OPTIONS**14498 The *iconv* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**. The  
14499 following options are supported:14500 **-f *fromcode***14501 Identify the codeset of the input file. Valid values for *fromcode* are specified in the  
14502 system documentation.14503 **-t *tocode***14504 Identify the codeset to be used for the output file. Valid values for *tocode* are specified  
14505 in the system documentation.14506 **OPERANDS**

14507 The following operands are supported:

14508 ***file*** A pathname of the input file to be translated. If *file* is omitted, the standard input is  
14509 used.14510 **STDIN**14511 The standard input is used only if the *file* operand is omitted.14512 **INPUT FILES**

14513 The input file is a text file.

14514 **ENVIRONMENT VARIABLES**14515 The following environment variables affect the execution of *iconv*:14516 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
14517 *LANG* is unset or null, the corresponding value from the implementation-dependent  
14518 default locale will be used. If any of the internationalisation variables contains an  
14519 invalid setting, the utility will behave as if none of the variables had been defined.14520 **LC\_ALL**14521 If set to a non-empty string value, override the values of all the other  
14522 internationalisation variables.14523 **LC\_CTYPE**14524 Determine the locale for the interpretation of sequences of bytes of text data as  
14525 characters (for example, single- as opposed to multi-byte characters in arguments).  
14526 During translation of the file, this variable is superseded by the use of the *fromcode*  
14527 option-argument.

14528 *LC\_MESSAGES*  
14529 Determine the locale that should be used to affect the format and contents of diagnostic  
14530 messages written to standard error.

14531 *NLSPATH*  
14532 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14533 **ASYNCHRONOUS EVENTS**  
14534 Default.

14535 **STDOUT**  
14536 The standard output is a text file containing the translated data.

14537 **STDERR**  
14538 Used only for diagnostic messages.

14539 **OUTPUT FILES**  
14540 None.

14541 **EXTENDED DESCRIPTION**  
14542 None.

14543 **EXIT STATUS**  
14544 The following exit values are returned:  
14545 0 Successful completion.  
14546 >0 An error occurred.

14547 **CONSEQUENCES OF ERRORS**  
14548 Default.

14549 **APPLICATION USAGE**  
14550 None.

14551 **EXAMPLES**  
14552 The following example converts the contents of file **mail.x400** from the ISO 6937:1983 standard  
14553 codeset to the ISO 8859-1:1987 standard codeset, and stores the results in file **mail.local**:  
14554 `iconv -f IS6937 -t IS8859 mail.x400 > mail.local`

14555 **FUTURE DIRECTIONS**  
14556 None.

14557 **SEE ALSO**  
14558 *gencat*.

14559 **CHANGE HISTORY**  
14560 First released in Issue 3.

14561 **Issue 4**  
14562 Format reorganised.  
14563 Utility Syntax Guidelines support mandated.  
14564 Internationalised environment variable support mandated.

14565 **NAME**14566        *id* — return user identity14567 **SYNOPSIS**14568        *id* [*user*]14569        *id* -G[-n] [*user*]14570        *id* -g[-nr] [*user*]14571        *id* -u[-nr] [*user*]14572 **DESCRIPTION**

14573        If no *user* operand is provided, the *id* utility will write the user and group IDs and the  
 14574        corresponding user and group names of the invoking process to standard output. If the effective  
 14575        and real IDs do not match, both will be written. If multiple groups are supported by the  
 14576        underlying system (see the description of {NGROUPS\_MAX} in the **XSH** specification), the  
 14577        supplementary group affiliations of the invoking process also will be written.

14578        If a *user* operand is provided and the process has the appropriate privileges, the user and group  
 14579        IDs of the selected user will be written. In this case, effective IDs will be assumed to be identical  
 14580        to real IDs. If the selected user has more than one allowable group membership listed in the  
 14581        group database, these will be written in the same manner as the supplementary groups  
 14582        described in the preceding paragraph.

14583 **OPTIONS**14584        The *id* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

14585        The following options are supported:

14586        **-G**        Output all different group IDs (effective, real and supplementary) only, using the  
 14587        format "%u\n". If there is more than one distinct group affiliation, output each such  
 14588        affiliation, using the format " %u", before the newline character is output.

14589        **-g**        Output only the effective group ID, using the format "%u\n".

14590        **-n**        Output the name in the format "%s" instead of the numeric ID using the format "%u".

14591        **-r**        Output the real ID instead of the effective ID.

14592        **-u**        Output only the effective user ID, using the format "%u\n".

14593 **OPERANDS**

14594        The following operand is supported:

14595        *user*       The login name for which information is to be written.

14596 **STDIN**

14597        Not used.

14598 **INPUT FILES**

14599        None.

14600 **ENVIRONMENT VARIABLES**14601        The following environment variables affect the execution of *id*:

14602        **LANG**       Provide a default value for the internationalisation variables that are unset or null. If  
 14603        **LANG** is unset or null, the corresponding value from the implementation-dependent  
 14604        default locale will be used. If any of the internationalisation variables contains an  
 14605        invalid setting, the utility will behave as if none of the variables had been defined.



14606 *LC\_ALL*  
 14607 If set to a non-empty string value, override the values of all the other  
 14608 internationalisation variables.

14609 *LC\_CTYPE*  
 14610 Determine the locale for the interpretation of sequences of bytes of text data as  
 14611 characters (for example, single- as opposed to multi-byte characters in arguments).

14612 *LC\_MESSAGES*  
 14613 Determine the locale that should be used to affect the format and contents of diagnostic  
 14614 messages written to standard error and informative messages written to standard  
 14615 output.

14616 EX *NLSPATH*  
 14617 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14618 **ASYNCHRONOUS EVENTS**  
 14619 Default.

14620 **STDOUT**  
 14621 The following formats will be used when the *LC\_MESSAGES* locale category specifies the  
 14622 POSIX locale. In other locales, the strings **uid**, **gid**, **euid**, **egid** and **groups** may be replaced with  
 14623 more appropriate strings corresponding to the locale.

14624 "uid=%u(%s) gid=%u(%s)\n", *<real user ID>*, *<user-name>*,  
 14625 *<real group ID>*, *<group-name>*

14626 If the effective and real user IDs do not match, the following will be inserted immediately before  
 14627 the \n character in the previous format:

14628 " euid=%u(%s) "

14629 with the following arguments added at the end of the argument list:

14630 *<effective user ID>*, *<effective user-name>*

14631 If the effective and real group IDs do not match, the following will be inserted directly before the  
 14632 \n character in the format string (and after any addition resulting from the effective and real  
 14633 user IDs not matching):

14634 " egid=%u(%s) "

14635 with the following arguments added at the end of the argument list:

14636 *<effective group-ID>*, *<effective group name>*

14637 If the process has supplementary group affiliations or the selected user is allowed to belong to  
 14638 multiple groups, the first will be added directly before the newline character in the format string:

14639 " groups=%u(%s) "

14640 with the following arguments added at the end of the argument list:

14641 *<supplementary group ID>*, *<supplementary group name>*

14642 and the necessary number of the following added after that for any remaining supplementary  
 14643 group IDs:

14644 " ,%u(%s) "

14645 and the necessary number of the following arguments added at the end of the argument list:

14646       <supplementary group ID>, <supplementary group name>

14647 If any of the user ID, group ID, effective user ID, effective group ID or supplementary/multiple  
 14648 group IDs cannot be mapped by the system into printable user or group names, the  
 14649 corresponding (%s) and name argument will be omitted from the corresponding format string.

14650 When any of the options are specified, the output format will be as described in the OPTIONS  
 14651 section.

14652 **STDERR**

14653       Used only for diagnostic messages.

14654 **OUTPUT FILES**

14655       None.

14656 **EXTENDED DESCRIPTION**

14657       None.

14658 **EXIT STATUS**

14659       The following exit values are returned:

14660       0   Successful completion.

14661       >0  An error occurred.

14662 **CONSEQUENCES OF ERRORS**

14663       Default.

14664 **APPLICATION USAGE**

14665       Output produced by the **-G** option and by the default case could potentially produce very long  
 14666 lines on systems that support large numbers of supplementary groups. (On systems with user  
 14667 and group IDs that are 32-bit integers and with group names with a maximum of 8 bytes per  
 14668 name, 93 supplementary groups plus distinct effective and real group and user IDs could  
 14669 theoretically overflow the 2048-byte {LINE\_MAX} text file line limit on the default output case.  
 14670 It would take about 186 supplementary groups to overflow the 2048-byte barrier using *id -G*).  
 14671 This is not expected to be a problem in practice, but in cases where it is a concern, applications  
 14672 should consider using *fold -s* before postprocessing the output of *id*.

14673 **EXAMPLES**

14674       None.

14675 **FUTURE DIRECTIONS**

14676       None.

14677 **SEE ALSO**

14678       *fold*, *logname*, *who*, the **XSH** specification description of *getgid()*, *getgroups()*, *getuid()*.

14679 **CHANGE HISTORY**

14680       First released in Issue 2.

14681 **Issue 4**

14682       Aligned with the ISO/IEC 9945-2: 1993 standard.

14683 **NAME**

14684       ipcrm — remove a message queue, semaphore set or shared memory segment identifier

14685 **SYNOPSIS**

```
14686 EX ipcrm [-q msgid | -Q msgkey | -s semid | -S semkey |
14687 -m shmid | -M shmkey] ...
```

14688

14689 **DESCRIPTION**14690       The *ipcrm* utility removes zero or more message queues, semaphore sets or shared memory  
14691       segments. The interprocess communication facilities to be removed are specified by the options.14692       Only a user with appropriate privilege is allowed to remove an interprocess communication  
14693       facility that was not created by or owned by the user invoking *ipcrm*.14694 **OPTIONS**14695       The *ipcrm* facility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**.

14696       The following options are supported:

14697       **-q msgid**14698           Remove the message queue identifier *msgid* from the system and destroy the message  
14699           queue and data structure associated with it.14700       **-m shmid**14701           Remove the shared memory identifier *shmid* from the system. The shared memory  
14702           segment and data structure associated with it are destroyed after the last detach.14703       **-s semid**14704           Remove the semaphore identifier *semid* from the system and destroy the set of  
14705           semaphores and data structure associated with it.14706       **-Q msgkey**14707           Remove the message queue identifier, created with key *msgkey*, from the system and  
14708           destroy the message queue and data structure associated with it.14709       **-M shmkey**14710           Remove the shared memory identifier, created with key *shmkey*, from the system. The  
14711           shared memory segment and data structure associated with it are destroyed after the  
14712           last detach.14713       **-S semkey**14714           Remove the semaphore identifier, created with key *semkey*, from the system and  
14715           destroy the set of semaphores and data structure associated with it.14716 **OPERANDS**

14717       None.

14718 **STDIN**

14719       Not used.

14720 **INPUT FILES**

14721       None.

14722 **ENVIRONMENT VARIABLES**14723       The following environment variables affect the execution of *ipcrm*:

14724       **LANG**   Provide a default value for the internationalization variables that are unset or null. If  
14725       **LANG** is unset or null, the corresponding value from the implementation-dependent  
14726       default locale will be used. If any of the internationalization variables contain an

|       |                                                                                                                 |
|-------|-----------------------------------------------------------------------------------------------------------------|
| 14727 | invalid setting, the utility will behave as if none of the variables had been set.                              |
| 14728 | <i>LC_ALL</i> If set to a non-empty string value, override the values of all the other                          |
| 14729 | internationalization variables.                                                                                 |
| 14730 | <i>LC_CTYPE</i>                                                                                                 |
| 14731 | Determine the locale for the interpretation of sequences of bytes of text data as                               |
| 14732 | characters (for example, single- as opposed to multi-byte characters in arguments).                             |
| 14733 | <i>LC_MESSAGES</i>                                                                                              |
| 14734 | Determine the locale that should be used to affect the format and contents of diagnostic                        |
| 14735 | messages written to standard error.                                                                             |
| 14736 | <i>NLSPATH</i>                                                                                                  |
| 14737 | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                         |
| 14738 | <b>ASYNCHRONOUS EVENTS</b>                                                                                      |
| 14739 | Default.                                                                                                        |
| 14740 | <b>STDOUT</b>                                                                                                   |
| 14741 | Not used.                                                                                                       |
| 14742 | <b>STDERR</b>                                                                                                   |
| 14743 | Used only for diagnostic messages.                                                                              |
| 14744 | <b>OUTPUT FILES</b>                                                                                             |
| 14745 | None.                                                                                                           |
| 14746 | <b>EXTENDED DESCRIPTION</b>                                                                                     |
| 14747 | None.                                                                                                           |
| 14748 | <b>EXIT STATUS</b>                                                                                              |
| 14749 | The following exit values are returned:                                                                         |
| 14750 | 0 Successful completion.                                                                                        |
| 14751 | >0 An error occurred.                                                                                           |
| 14752 | <b>CONSEQUENCES OF ERRORS</b>                                                                                   |
| 14753 | Default.                                                                                                        |
| 14754 | <b>APPLICATION USAGE</b>                                                                                        |
| 14755 | None.                                                                                                           |
| 14756 | <b>EXAMPLES</b>                                                                                                 |
| 14757 | None.                                                                                                           |
| 14758 | <b>FUTURE DIRECTIONS</b>                                                                                        |
| 14759 | None.                                                                                                           |
| 14760 | <b>SEE ALSO</b>                                                                                                 |
| 14761 | <i>ipcs</i> , the <b>XSH</b> specification description of <i>msgctl()</i> , <i>semctl()</i> , <i>shmctl()</i> . |
| 14762 | <b>CHANGE HISTORY</b>                                                                                           |
| 14763 | First released in Issue 5.                                                                                      |

14764 **NAME**

14765            *ipcs* — report inter-process communication facilities status

14766 **SYNOPSIS**

14767 EX        *ipcs* [ *-qms* ] [ *-abcopt* ]

14768

14769 **DESCRIPTION**

14770            The *ipcs* utility writes information about active inter-process communication facilities.

14771            Without options, information is written in short format for message queues, shared memory  
 14772 segments and semaphores sets that are currently active in the system. Otherwise, the  
 14773 information that is displayed is controlled by the options specified.

14774 **OPTIONS**

14775            The *ipcs* facility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

14776            The *ipcs* utility accepts the following options:

14777            **-q**        Write information about active message queues.

14778            **-m**        Write information about active shared memory segments.

14779            **-s**        Write information about active semaphores sets.

14780            If **-q**, **-m** or **-s** are specified, only information about those facilities is written. If none of these  
 14781 three are specified, information about all three is written subject to the following options:

14782            **-a**        Use all print options. (This is a shorthand notation for **-b**, **-c**, **-o**, **-p** and **-t**.)

14783            **-b**        Write information on maximum allowable size. (Maximum number of bytes in  
 14784 messages on queue for message queues, size of segments for shared memory, and  
 14785 number of semaphores in each set for semaphores.)

14786            **-c**        Write creator's user name and group name. See below.

14787            **-o**        Write information on outstanding usage. (Number of messages on queue and total  
 14788 number of bytes in messages on queue for message queues, and number of processes  
 14789 attached to shared memory segments.)

14790            **-p**        Write process number information. (Process ID of last process to send a message and  
 14791 process ID of last process to receive a message on message queues, process ID of  
 14792 creating process, and process ID of last process to attach or detach on shared memory  
 14793 segments.)

14794            **-t**        Write time information. (Time of the last control operation that changed the access  
 14795 permissions for all facilities, time of last *msgsnd()* and *msgrcv()* operations on message  
 14796 queues, time of last *shmat()* and *shmdt()* operations on shared memory, and time of last  
 14797 *semop()* operation on semaphores.)

14798 **OPERANDS**

14799            None.

14800 **STDIN**

14801            Not used.

14802 **INPUT FILES**

14803            • the group database

14804            • the user database.

## 14805 ENVIRONMENT VARIABLES

14806 The following environment variables affect the execution of *ipcs*:

14807 *LANG* Provide a default value for the internationalization variables that are unset or null. If  
14808 *LANG* is unset or null, the corresponding value from the implementation-dependent  
14809 default locale will be used. If any of the internationalization variables contain an  
14810 invalid setting, the utility will behave as if none of the variables had been set.

14811 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
14812 internationalization variables.

14813 *LC\_CTYPE*

14814 Determine the locale for the interpretation of sequences of bytes of text data as  
14815 characters (for example, single- as opposed to multi-byte characters in arguments).

14816 *LC\_MESSAGES*

14817 Determine the locale that should be used to affect the format and contents of diagnostic  
14818 messages written to standard error.

14819 *NLSPATH*

14820 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

14821 *TZ* Determine the timezone for the time strings written by *ipcs*.

## 14822 ASYNCHRONOUS EVENTS

14823 Default.

## 14824 STDOUT

14825 An introductory line is written with the format:

14826 "IPC status from %s as of %s\n", <source>, <date>

14827 where <source> indicates the source used to gather the statistics and <date> is the information  
14828 that would be produced by the command:

14829 date

14830 when invoked in the POSIX locale.

14831 The *ipcs* utility then creates up to three reports depending upon the *-q*, *-m* and *-s* options. The  
14832 first report indicates the status of message queues, the second report indicates the status of  
14833 shared memory segments, and the third report indicates the status of semaphore sets.

14834 If the corresponding facility is not installed or has not been used since the last reboot, then the  
14835 report is written out in the format:

14836 "%s facility not in system.\n", <facility>

14837 where <facility> is Message Queue, Shared Memory or Semaphore as appropriate. If the facility  
14838 has been installed and has been used since the last reboot, column headings separated by one or  
14839 more spaces and followed by a new line will be written as indicated below and the facility name  
14840 is written out using the format:

14841 "%s:\n", <facility>

14842 where <facility> is Message Queues, Shared Memory or Semaphores as appropriate. On the  
14843 second and third reports the column headings need not be written if the last column headings  
14844 written already provide column headings for all information in that report.

14845 The column headings provided in the first column below and the meaning of the information in  
14846 those columns are given in order below; the letters in parentheses indicate the options that cause  
14847 the corresponding column to appear; "all" means that the column always appears. Each column

|       |                                                                                           |               |                                                                                                   |
|-------|-------------------------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------------------|
| 14848 | is separated by one or more space characters. Note that these options only determine what |               |                                                                                                   |
| 14849 | information is provided for each report; they do not determine which reports are written. |               |                                                                                                   |
| 14850 | T                                                                                         | (all)         | Type of facility:                                                                                 |
| 14851 |                                                                                           | <b>q</b>      | Message queue.                                                                                    |
| 14852 |                                                                                           | <b>m</b>      | Shared memory segment.                                                                            |
| 14853 |                                                                                           | <b>s</b>      | Semaphore.                                                                                        |
| 14854 | This field is a single character written using the format "%c".                           |               |                                                                                                   |
| 14855 | ID                                                                                        | (all)         | The identifier for the facility entry. This field is written using the format "%d".               |
| 14856 | KEY                                                                                       | (all)         | The key used as an argument to <i>msgget()</i> , <i>semget()</i> or <i>shmget()</i> to create the |
| 14857 |                                                                                           |               | facility entry.                                                                                   |
| 14858 |                                                                                           | <b>Note:</b>  | The key of a shared memory segment is changed to IPC_PRIVATE                                      |
| 14859 |                                                                                           |               | when the segment has been removed until all processes attached to                                 |
| 14860 |                                                                                           |               | the segment detach it.                                                                            |
| 14861 | This field is written using the format "0x%x".                                            |               |                                                                                                   |
| 14862 | MODE                                                                                      | (all)         | The facility access modes and flags. The mode consists of 11 characters that                      |
| 14863 |                                                                                           |               | are interpreted as follows.                                                                       |
| 14864 | The first character is:                                                                   |               |                                                                                                   |
| 14865 |                                                                                           | <b>S</b>      | If a process is waiting on a <i>msgsnd()</i> operation.                                           |
| 14866 |                                                                                           | –             | If the above is not true.                                                                         |
| 14867 | The second character is:                                                                  |               |                                                                                                   |
| 14868 |                                                                                           | <b>R</b>      | If a process is waiting on a <i>msgrcv()</i> operation.                                           |
| 14869 |                                                                                           | <b>C</b> or - | If the associated shared memory segment is to be cleared when the                                 |
| 14870 |                                                                                           |               | first attach operation is executed.                                                               |
| 14871 |                                                                                           | –             | If none of the above is true.                                                                     |
| 14872 | The next nine characters are interpreted as three sets of three bits each. The            |               |                                                                                                   |
| 14873 | first set refers to the owner's permissions; the next to permissions of others in         |               |                                                                                                   |
| 14874 | the usergroup of the facility entry; and the last to all others. Within each set,         |               |                                                                                                   |
| 14875 | the first character indicates permission to read, the second character indicates          |               |                                                                                                   |
| 14876 | permission to write or alter the facility entry, and the last character is a minus        |               |                                                                                                   |
| 14877 | sign (–).                                                                                 |               |                                                                                                   |
| 14878 | The permissions are indicated as follows:                                                 |               |                                                                                                   |
| 14879 |                                                                                           | <b>r</b>      | If read permission is granted.                                                                    |
| 14880 |                                                                                           | <b>w</b>      | If write permission is granted.                                                                   |
| 14881 |                                                                                           | <b>a</b>      | If alter permission is granted.                                                                   |
| 14882 |                                                                                           | –             | If the indicated permission is not granted.                                                       |
| 14883 | The first character following the permissions specifies if there is an alternate          |               |                                                                                                   |
| 14884 | or additional access control method associated with the facility. If there is no          |               |                                                                                                   |
| 14885 | alternate or additional access control method associated with the facility, a             |               |                                                                                                   |
| 14886 | single space character will be written; otherwise another printable character             |               |                                                                                                   |
| 14887 | will be written.                                                                          |               |                                                                                                   |

|       |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
|-------|---------------------------------------------------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14888 | OWNER                                                               | (all) | The user name of the owner of the facility entry. If the user name of the owner is found in the user database, at least the first eight column positions of the name are written using the format "%s". Otherwise, the user ID of the owner is written using the format "%d".                                             |
| 14889 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14890 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14891 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14892 | GROUP                                                               | (all) | The group name of the owner of the facility entry. If the group name of the owner is found in the group database, at least the first eight column positions of the name are written using the format "%s". Otherwise, the group ID of the owner is written using the format "%d".                                         |
| 14893 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14894 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14895 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14896 | The following nine columns are only written out for message queues: |       |                                                                                                                                                                                                                                                                                                                           |
| 14897 | CREATOR                                                             | (a,c) | The user name of the creator of the facility entry. If the user name of the creator is found in the user database, at least the first eight column positions of the name are written using the format "%s". Otherwise, the user ID of the creator is written using the format "%d".                                       |
| 14898 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14899 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14900 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14901 | CGROUP                                                              | (a,c) | The group name of the creator of the facility entry. If the group name of the creator is found in the group database, at least the first eight column positions of the name are written using the format "%s". Otherwise, the group ID of the creator is written using the format "%d".                                   |
| 14902 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14903 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14904 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14905 | CBYTES                                                              | (a,o) | The number of bytes in messages currently outstanding on the associated message queue. This field is written using the format "%d".                                                                                                                                                                                       |
| 14906 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14907 | QNUM                                                                | (a,o) | The number of messages currently outstanding on the associated message queue. This field is written using the format "%d".                                                                                                                                                                                                |
| 14908 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14909 | QBYTES                                                              | (a,b) | The maximum number of bytes allowed in messages outstanding on the associated message queue. This field is written using the format "%d".                                                                                                                                                                                 |
| 14910 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14911 | LSPID                                                               | (a,p) | The process ID of the last process to send a message to the associated queue. This field is written using the format:                                                                                                                                                                                                     |
| 14912 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14913 |                                                                     |       | "%d", <pid>                                                                                                                                                                                                                                                                                                               |
| 14914 |                                                                     |       | where <pid> is 0 if no message has been sent to the corresponding message queue; otherwise, <pid> is the process ID of the last process to send a message to the queue.                                                                                                                                                   |
| 14915 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14916 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14917 | LRPID                                                               | (a,p) | The process ID of the last process to receive a message from the associated queue. This field is written using the format:                                                                                                                                                                                                |
| 14918 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14919 |                                                                     |       | "%d", <pid>                                                                                                                                                                                                                                                                                                               |
| 14920 |                                                                     |       | where <pid> is 0 if no message has been received from the corresponding message queue; otherwise, <pid> is the process ID of the last process to receive a message from the queue.                                                                                                                                        |
| 14921 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14922 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14923 | STIME                                                               | (a,t) | The time the last message was sent to the associated queue. If a message has been sent to the corresponding message queue, the hour, minute and second of the last time a message was sent to the queue is written using the format "%d:%2.2d:%2.2d". Otherwise, the format "no-entry" will be written.                   |
| 14924 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14925 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14926 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14927 | RTIME                                                               | (a,t) | The time the last message was received from the associated queue. If a message has been received from the corresponding message queue, the hour, minute and second of the last time a message was received from the queue is written using the format "%d:%2.2d:%2.2d". Otherwise, the format "no-entry" will be written. |
| 14928 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14929 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14930 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |
| 14931 |                                                                     |       |                                                                                                                                                                                                                                                                                                                           |



|       |                                                                             |       |                                                                                 |
|-------|-----------------------------------------------------------------------------|-------|---------------------------------------------------------------------------------|
| 14932 | The following six columns are only written out for shared memory segments.  |       |                                                                                 |
| 14933 | NATTCH                                                                      | (a,o) | The number of processes attached to the associated shared memory segment.       |
| 14934 |                                                                             |       | This field is written using the format "%d".                                    |
| 14935 | SEGSZ                                                                       | (a,b) | The size of the associated shared memory segment. This field is written using   |
| 14936 |                                                                             |       | the format "%d".                                                                |
| 14937 | CPID                                                                        | (a,p) | The process ID of the creator of the shared memory entry. This field is written |
| 14938 |                                                                             |       | using the format "%d".                                                          |
| 14939 | LPID                                                                        | (a,p) | The process ID of the last process to attach or detach the shared memory        |
| 14940 |                                                                             |       | segment. This field is written using the format:                                |
| 14941 |                                                                             |       | "%d", <pid>                                                                     |
| 14942 |                                                                             |       | where <pid> is 0 if no process has attached the corresponding shared memory     |
| 14943 |                                                                             |       | segment; otherwise, <pid> is the process ID of the last process to attach or    |
| 14944 |                                                                             |       | detach the segment.                                                             |
| 14945 | ATIME                                                                       | (a,t) | The time the last attach on the associated shared memory segment was            |
| 14946 |                                                                             |       | completed. If the corresponding shared memory segment has ever been             |
| 14947 |                                                                             |       | attached, the hour, minute and second of the last time the segment was          |
| 14948 |                                                                             |       | attached is written using the format "%d:%2.2d:%2.2d". Otherwise, the format    |
| 14949 |                                                                             |       | " no-entry" will be written.                                                    |
| 14950 | DTIME                                                                       | (a,t) | The time the last detach on the associated shared memory segment was            |
| 14951 |                                                                             |       | completed. If the corresponding shared memory segment has ever been             |
| 14952 |                                                                             |       | detached, the hour, minute, and second of the last time the segment was         |
| 14953 |                                                                             |       | detached is written using the format "%d:%2.2d:%2.2d". Otherwise, the           |
| 14954 |                                                                             |       | format " no-entry" will be written.                                             |
| 14955 | The following two columns are only written out for semaphore sets:          |       |                                                                                 |
| 14956 | NSEMS                                                                       | (a,t) | The number of semaphores in the set associated with the semaphore entry.        |
| 14957 |                                                                             |       | This field is written using the format "%d".                                    |
| 14958 | OTIME                                                                       | (a,t) | The time the last semaphore operation on the set associated with the            |
| 14959 |                                                                             |       | semaphore entry was completed. If a semaphore operation has ever been           |
| 14960 |                                                                             |       | performed on the corresponding semaphore set, the hour, minute and second       |
| 14961 |                                                                             |       | of the last semaphore operation on the semaphore set is written using the       |
| 14962 |                                                                             |       | format "%d:%2.2d:%2.2d". Otherwise, the format " no-entry" will be              |
| 14963 |                                                                             |       | written.                                                                        |
| 14964 | The following column is written for all three reports when it is requested: |       |                                                                                 |
| 14965 | CTIME                                                                       | (a,t) | The time the associated entry was created or changed. The hour, minute and      |
| 14966 |                                                                             |       | second of the time when the associated entry was created are written using      |
| 14967 |                                                                             |       | the format "%d:%2.2d:%2.2d".                                                    |
| 14968 | <b>STDERR</b>                                                               |       |                                                                                 |
| 14969 | Used only for diagnostic messages.                                          |       |                                                                                 |
| 14970 | <b>OUTPUT FILES</b>                                                         |       |                                                                                 |
| 14971 | None.                                                                       |       |                                                                                 |
| 14972 | <b>EXTENDED DESCRIPTION</b>                                                 |       |                                                                                 |
| 14973 | None.                                                                       |       |                                                                                 |

14974 **EXIT STATUS**

14975           The following exit values are returned:

14976           0   Successful completion.

14977           >0  An error occurred.

14978 **CONSEQUENCES OF ERRORS**

14979           Default.

14980 **APPLICATION USAGE**

14981           Things can change while *ipcs* is running; the information it gives is guaranteed to be accurate  
14982           only when it was retrieved.

14983 **EXAMPLES**

14984           None.

14985 **FUTURE DIRECTIONS**

14986           None.

14987 **SEE ALSO**

14988           The **XSH** specification description of *msgop()*, *msgrcv()*, *msgsnd()*, *semget()*, *semop()*, *shmat()*,  
14989           *shmdt()*, *shmget()*, *shmop()*.

14990 **CHANGE HISTORY**

14991           First released in Issue 5.

14992 **NAME**14993       *jobs* — display status of jobs in the current session14994 **SYNOPSIS**14995 **JC**       *jobs* [ *-l* | *-p* ][ *job\_id...* ]14996 **DESCRIPTION**14997       The *jobs* utility displays the status of jobs that were started in the current shell environment; see  
14998       Section 2.12 on page 63.14999       When *jobs* reports the termination status of a job, the shell removes its process ID from the list of  
15000       those “known in the current shell execution environment”; see Section 2.9.3 on page 50.15001 **OPTIONS**15002       The *jobs* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**.

15003       The following options are supported:

15004       **-l**       (The letter ell.) Provide more information about each job listed. This information  
15005       includes the job number, current job, process group ID, state and the command that  
15006       formed the job.15007       **-p**       Display only the process IDs for the process group leaders of the selected jobs.15008       By default, the *jobs* utility displays the status of all stopped jobs, running background jobs and  
15009       all jobs whose status has changed and have not been reported by the shell.15010 **OPERANDS**

15011       The following operand is supported:

15012       *job\_id*   Specifies the jobs for which the status is to be displayed. If no *job\_id* is given, the status  
15013       information for all jobs will be displayed. The format of *job\_id* is described in the entry  
15014       for **job control job ID** in the **XBD specification, Chapter 2, Glossary**.15015 **STDIN**

15016       Not used.

15017 **INPUT FILES**

15018       None.

15019 **ENVIRONMENT VARIABLES**15020       The following environment variables affect the execution of *jobs*:15021       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
15022       **LANG** is unset or null, the corresponding value from the implementation-dependent  
15023       default locale will be used. If any of the internationalisation variables contains an  
15024       invalid setting, the utility will behave as if none of the variables had been defined.15025       **LC\_ALL**15026       If set to a non-empty string value, override the values of all the other  
15027       internationalisation variables.15028       **LC\_CTYPE**15029       Determine the locale for the interpretation of sequences of bytes of text data as  
15030       characters (for example, single- as opposed to multi-byte characters in arguments).15031       **LC\_MESSAGES**15032       Determine the locale that should be used to affect the format and contents of diagnostic  
15033       messages written to standard error and informative messages written to standard  
15034       output.

15035 EX **NLSPATH**  
 15036 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

15037 **ASYNCHRONOUS EVENTS**  
 15038 Default.

15039 **STDOUT**  
 15040 If the **-p** option is specified, the output consists of one line for each process ID:  
 15041 "%d\n", *<process ID>*  
 15042 Otherwise, if the **-l** option is not specified, the output is a series of lines of the form:  
 15043 "[%d] %c %s %s\n", *<job-number>*, *<current>*, *<state>*, *<command>*  
 15044 where the fields are as follows:

15045 **<current>**  
 15046 The character "+" identifies the job that would be used as a default for the *fg* or *bg*  
 15047 utilities; this job can also be specified using the *job\_id* %+ or %%. The character "-"  
 15048 identifies the job that would become the default if the current default job were to exit;  
 15049 this job can also be specified using the *job\_id* %-. For other jobs, this field is a space  
 15050 character. At most one job can be identified with "+" and at most one job can be  
 15051 identified with "-". If there is any suspended job, then the current job will be a  
 15052 suspended job. If there are at least two suspended jobs, then the previous job will also  
 15053 be a suspended job.

15054 **<job-number>**  
 15055 A number that can be used to identify the process group to the *wait*, *fg*, *bg* and *kill*  
 15056 utilities. Using these utilities, the job can be identified by prefixing the job number with  
 15057 "%".

15058 **<state>** One of the following strings (in the POSIX locale):

15059 **Running**  
 15060 Indicates that the job has not been suspended by a signal and has not exited.

15061 **Done** Indicates that the job completed and returned exit status zero.

15062 **Done(*code*)**  
 15063 Indicates that the job completed normally and that it exited with the specified  
 15064 non-zero exit status, *code*, expressed as a decimal number.

15065 **Stopped**  
 15066 **Stopped (SIGTSTP)**  
 15067 Indicates that the job was suspended by the SIGTSTP signal.

15068 **Stopped (SIGSTOP)**  
 15069 Indicates that the job was suspended by the SIGSTOP signal.

15070 **Stopped (SIGTTIN)**  
 15071 Indicates that the job was suspended by the SIGTTIN signal.

15072 **Stopped (SIGTTOU)**  
 15073 Indicates that the job was suspended by the SIGTTOU signal.

15074 The implementation may substitute the string **Suspended** in place of **Stopped**. If the  
 15075 job was terminated by a signal, the format of *<state>* is unspecified, but it will be visibly  
 15076 distinct from all of the other *<state>* formats shown here and will indicate the name or  
 15077 description of the signal causing the termination.

15078           <command>  
15079           The associated command that was given to the shell.

15080           If the **-l** option is specified, a field containing the process group ID is inserted before the <state>  
15081           field. Also, more processes in a process group may be output on separate lines, using only the  
15082           process ID and <command> fields.

15083 **STDERR**  
15084           Used only for diagnostic messages.

15085 **OUTPUT FILES**  
15086           None.

15087 **EXTENDED DESCRIPTION**  
15088           None.

15089 **EXIT STATUS**  
15090           The following exit values are returned:  
15091           0   Successful completion.  
15092           >0  An error occurred.

15093 **CONSEQUENCES OF ERRORS**  
15094           Default.

15095 **APPLICATION USAGE**  
15096           The **-p** option is the only portable way to find out the process group of a job because different  
15097           implementations have different strategies for defining the process group of the job. Usage such  
15098           as \$(jobs -p) provides a way of referring to the process group of the job in an implementation-  
15099           independent way.

15100           The *jobs* utility will not work as expected when it is operating in its own utility execution  
15101           environment because that environment will have no applicable jobs to manipulate. See the  
15102           APPLICATION USAGE section for *bg*. For this reason, *jobs* is generally implemented as a shell  
15103           regular built-in.

15104 **EXAMPLES**  
15105           None.

15106 **FUTURE DIRECTIONS**  
15107           None.

15108 **SEE ALSO**  
15109           *bg*, *fg*, *kill*, *wait*.

15110 **CHANGE HISTORY**  
15111           First released in Issue 4.

## 15112 NAME

15113 join — relational database operator

## 15114 SYNOPSIS

```
15115 join [-a file_number | -v file_number] [-e string] [-o list] [-t char]
15116 [-1 field] [-2 field] file1 file2
```

```
15117 OB join [-a file_number] [-e string] [-j field] [-j1 field] [-j2 field]
15118 [-o list...] [-t char] [-t char] file1 file2
```

## 15119 DESCRIPTION

15120 The *join* utility will perform an equality join on the files *file1* and *file2*. The joined files will be  
 15121 written to the standard output.

15122 The join field is a field in each file on which the files are compared. By default, *join* writes one  
 15123 line in the output for each pair of lines in *file1* and *file2* that have identical join fields. The output  
 15124 line by default will consist of the join field, then the remaining fields from *file1*, then the  
 15125 remaining fields from *file2*. This format can be changed by using the **-o** option (see below). The  
 15126 **-a** option can be used to add unmatched lines to the output. The **-v** option can be used to  
 15127 output only unmatched lines.

15128 By default, the files *file1* and *file2* should be ordered in the collating sequence of *sort -b* on the  
 15129 fields on which they are to be joined, by default the first in each line. All selected output will be  
 15130 written in the same collating sequence.

15131 The default input field separators will be blank characters. In this case, multiple separators will  
 15132 count as one field separator, and leading separators will be ignored. The default output field  
 15133 separator will be a space character.

15134 The field separator and collating sequence can be changed by using the **-t** option (see below).

15135 If the input files are not in the appropriate collating sequence, the results are unspecified.

## 15136 OPTIONS

15137 OB The *join* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 15138 obsolescent version does not follow the utility argument syntax guidelines: the **-j1** and **-j2**  
 15139 options are multi-character options and the **-o** option takes multiple arguments.

15140 The following options are supported:

15141 **-a file\_number**

15142 Produce a line for each unpairable line in file *file\_number*, where *file\_number* is 1 or 2, in  
 15143 addition to the default output. If both **-a 1** and **-a 2** are specified, all unpairable lines  
 15144 will be output.

15145 **-e string**

15146 Replace empty output fields in the list selected by **-o** with the string *string*.

15147 OB **-j field** Equivalent to: **-1 field -2 field**.

15148 OB **-j1 field** Equivalent to: **-1 field**.

15149 OB **-j2 field** Equivalent to: **-2 field**.

15150 **-o list** Construct the output line to comprise the fields specified in *list*, each element of which  
 15151 has one of the following two forms:

15152 — *file\_number.field*, where *file\_number* is a file number and *field* is a decimal integer field  
 15153 number

15154 — 0 (zero), representing the join field.

15155 The elements of *list* are either comma- or blank-separated, as specified in Guideline 8 of  
 15156 the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The fields specified by  
 15157 *list* will be written for all selected output lines. Fields selected by *list* that do not appear  
 15158 in the input will be treated as empty output fields. (See the **-e** option.) Only  
 15159 specifically requested fields are written. The *list* must be a single command line  
 15160 OB argument. However, as an obsolescent feature, the argument *list* can be multiple  
 15161 arguments on the command line.

15162 **-t char** Use character *char* as a separator, for both input and output. Every appearance of *char*  
 15163 in a line will be significant. When this option is specified, the collating sequence should  
 15164 be the same as *sort* without the **-b** option.

15165 **-v file\_number**  
 15166 Instead of the default output, produce a line only for each unpairable line in  
 15167 *file\_number*, where *file\_number* is 1 or 2. If both **-v 1** and **-v 2** are specified, all  
 15168 unpairable lines will be output.

15169 **-1 field** Join on the *fieldth* field of file 1. Fields are decimal integers starting with 1.

15170 **-2 field** Join on the *fieldth* field of file 2. Fields are decimal integers starting with 1.

15171 **OPERANDS**  
 15172 The following operands are supported:

15173 *file1*  
 15174 *file2* A pathname of a file to be joined. If either of the *file1* or *file2* operands is "-", the  
 15175 standard input is used in its place.

15176 **STDIN**  
 15177 The standard input will be used only if the *file1* or *file2* operand is "-". See the INPUT FILES  
 15178 section.

15179 **INPUT FILES**  
 15180 The input files must be text files.

15181 **ENVIRONMENT VARIABLES**  
 15182 The following environment variables affect the execution of *join*:

15183 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 15184 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 15185 default locale will be used. If any of the internationalisation variables contains an  
 15186 invalid setting, the utility will behave as if none of the variables had been defined.

15187 **LC\_ALL**  
 15188 If set to a non-empty string value, override the values of all the other  
 15189 internationalisation variables.

15190 **LC\_COLLATE**  
 15191 Determine the locale of the collating sequence *join* expects to have been used when the  
 15192 input files were sorted.

15193 **LC\_CTYPE**  
 15194 Determine the locale for the interpretation of sequences of bytes of text data as  
 15195 characters (for example, single- as opposed to multi-byte characters in arguments and  
 15196 input files).

15197 **LC\_MESSAGES**  
 15198 Determine the locale that should be used to affect the format and contents of diagnostic

15199                    messages written to standard error.

15200 EX            **NLSPATH**

15201                    Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 15202 ASYNCHRONOUS EVENTS

15203                    Default.

## 15204 STDOUT

15205                    The *join* utility output will be a concatenation of selected character fields. When the **-o** option is  
15206                    not specified, the output will be:

15207                    "*%s%s%s\n*", *<join field>*, *<other file1 fields>*,  
15208                    *<other file2 fields>*

15209                    If the join field is not the first field in a file, the *<other file fields>* for that file are:

15210                    *<fields preceding join field>*, *<fields following join field>*

15211                    When the **-o** option is specified, the output format will be:

15212                    "*%s\n*", *<concatenation of fields>*

15213                    where the concatenation of fields is described by the **-o** option, above.

15214                    For either format, each field (except the last) will be written with its trailing separator character.  
15215                    If the separator is the default (blank characters), a single space character will be written after  
15216                    each field (except the last).

## 15217 STDERR

15218                    Used only for diagnostic messages.

## 15219 OUTPUT FILES

15220                    None.

## 15221 EXTENDED DESCRIPTION

15222                    None.

## 15223 EXIT STATUS

15224                    The following exit values are returned:

15225                    0    All input files were output successfully.

15226                    >0   An error occurred.

## 15227 CONSEQUENCES OF ERRORS

15228                    Default.

## 15229 APPLICATION USAGE

15230                    Pathnames consisting of numeric digits or of the form *string.string* should not be specified  
15231                    directly following the **-o** list.

## 15232 EXAMPLES

15233                    The **-o 0** field essentially selects the union of the join fields. For example, given file **phone**:

|       |         |                 |
|-------|---------|-----------------|
| 15234 | !Name   | Phone Number    |
| 15235 | Don     | +1 123-456-7890 |
| 15236 | Hal     | +1 234-567-8901 |
| 15237 | Yasushi | +2 345-678-9012 |



15238 and file **fax**:

|       |         |                 |
|-------|---------|-----------------|
| 15239 | !Name   | Fax Number      |
| 15240 | Don     | +1 123-456-7899 |
| 15241 | Keith   | +1 456-789-0122 |
| 15242 | Yasushi | +2 345-678-9011 |

15243 (where the large expanses of white space are meant to each represent a single tab character), the  
 15244 command:

15245 `join -t "<tab>" -a 1 -a 2 -e '(unknown)' -o 0,1.2,2.2 phone fax`

15246 would produce:

|       |         |                 |                 |
|-------|---------|-----------------|-----------------|
| 15247 | !Name   | Phone Number    | Fax Number      |
| 15248 | Don     | +1 123-456-7890 | +1 123-456-7899 |
| 15249 | Hal     | +1 234-567-8901 | (unknown)       |
| 15250 | Keith   | (unknown)       | +1 456-789-0122 |
| 15251 | Yasushi | +2 345-678-9012 | +2 345-678-9011 |

#### 15252 FUTURE DIRECTIONS

15253 The obsolescent **-j** options and the multi-argument **-o** option may be withdrawn in a future  
 15254 issue. |

#### 15255 SEE ALSO

15256 *awk, comm, sort, uniq.*

#### 15257 CHANGE HISTORY

15258 First released in Issue 2. |

#### 15259 Issue 4

15260 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 15261 NAME

15262 kill — terminate or signal processes

## 15263 SYNOPSIS

15264 kill -s *signal\_name* *pid*...15265 kill -l [*exit\_status*]15266 OB kill [-*signal\_name*] *pid*...15267 OB kill [-*signal\_number*] *pid*...

## 15268 DESCRIPTION

15269 The *kill* utility will send a signal to the process or processes specified by each *pid* operand.15270 For each *pid* operand, the *kill* utility will perform actions equivalent to the **XSH** specification  
15271 *kill()* function called with the following arguments:

- 15272 1. The value of the *pid* operand will be used as the *pid* argument.
- 15273 2. The *sig* argument is the value specified by the **-s** option, **-signal\_number** option, or the
- 15274 **-signal\_name** option, or by SIGTERM, if none of these options is specified.

## 15275 OPTIONS

15276 OB The *kill* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 15277 that in the obsolescent form, the **-signal\_number** and **-signal\_name** options are usually more than  
 15278 a single character.

15279 The following options are supported:

15280 **-l** (The letter ell.) Write all values of *signal\_name* supported by the implementation, if no  
 15281 operand is given. If an *exit\_status* operand is given and it is a value of the "?" shell  
 15282 special parameter (see Section 2.5.2 on page 27 and *wait*) corresponding to a process  
 15283 that was terminated by a signal, the *signal\_name* corresponding to the signal that  
 15284 terminated the process will be written. If an *exit\_status* operand is given and it is the  
 15285 unsigned decimal integer value of a signal number, the *signal\_name* (the **XSH**  
 15286 specification-defined symbolic constant name without the **SIG** prefix) corresponding to  
 15287 that signal will be written. Otherwise, the results are unspecified.

15288 **-s signal\_name**

15289 Specify the signal to send, using one of the symbolic names defined in the **XSH**  
 15290 specification <**signal.h**> description. Values of *signal\_name* will be recognised in a  
 15291 case-independent fashion, without the **SIG** prefix. In addition, the symbolic name 0  
 15292 will be recognised, representing the signal value zero. The corresponding signal will be  
 15293 sent instead of SIGTERM.

15294 OB **-signal\_name**15295 Equivalent to **-s signal\_name**.

15296 OB

*-signal\_number*

15297

Specify a non-negative decimal integer, *signal\_number*, representing the signal to be used instead of SIGTERM, as the *sig* argument in the effective call to *kill()*. The correspondence between integer values and the *sig* value used is shown in the following table.

15298

15299

15300

15301

15302

| <i>signal_number</i> | <i>sig Value</i> |
|----------------------|------------------|
| 0                    | 0                |
| 1                    | SIGHUP           |
| 2                    | SIGINT           |
| 3                    | SIGQUIT          |
| 6                    | SIGABRT          |
| 9                    | SIGKILL          |
| 14                   | SIGALRM          |
| 15                   | SIGTERM          |

15303

15304

15305

15306

15307

15308

15309

15310

15311

The effects of specifying any *signal\_number* other than those listed in the table are undefined.

15312

15313

In the obsolescent versions, if the first argument is a negative integer, it will be interpreted as a *-signal\_number* option, not as a negative *pid* operand specifying a process group.

15314

15315 **OPERANDS**

15316

The following operands are supported:

15317

*pid* One of the following:

15318

1. A decimal integer specifying a process or process group to be signalled. The process or processes selected by positive, negative and zero values of the *pid* operand will be as described for the **XSH** specification *kill()* function. If process number 0 is specified, all processes in the process group are signalled. For the effects of negative *pid* numbers, see the **XSH** specification under *kill()*. If the first *pid* operand is negative, it should be preceded by *--* to keep it from being interpreted as an option.

15319

15320

15321

15322

15323

15324

15325

2. A job control job ID (see the **XBD** specification, **Chapter 2, Glossary**) that identifies a background process group to be signalled. The job control job ID notation is applicable only for invocations of *kill* in the current shell execution environment; see Section 2.12 on page 63.

15326

15327

15328

15329

**Note:** The job control job ID type of *pid* is available on systems supporting both the job control option and the User Portability Utilities Option, which applies to all XSI-conformant systems.

15330

15331

15332

*exit\_status*

15333

A decimal integer specifying a signal number or the exit status of a process terminated by a signal.

15334

15335 **STDIN**

15336

Not used.

15337 **INPUT FILES**

15338

None.

15339 **ENVIRONMENT VARIABLES**

15340 The following environment variables affect the execution of *kill*:

15341 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 15342 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 15343 default locale will be used. If any of the internationalisation variables contains an  
 15344 invalid setting, the utility will behave as if none of the variables had been defined.

15345 **LC\_ALL**

15346 If set to a non-empty string value, override the values of all the other  
 15347 internationalisation variables.

15348 **LC\_CTYPE**

15349 Determine the locale for the interpretation of sequences of bytes of text data as  
 15350 characters (for example, single- as opposed to multi-byte characters in arguments).

15351 **LC\_MESSAGES**

15352 Determine the locale that should be used to affect the format and contents of diagnostic  
 15353 messages written to standard error.

15354 EX **NLSPATH**

15355 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

15356 **ASYNCHRONOUS EVENTS**

15357 Default.

15358 **STDOUT**

15359 When the **-l** option is not specified, the standard output will not be used.

15360 When the **-l** option is specified, the symbolic name of each signal will be written in the following  
 15361 format:

15362 "%s%c", <signal\_name>, <separator>

15363 where the <signal\_name> is in upper-case, without the **SIG** prefix, and the <separator> will be  
 15364 either a newline character or a space character. For the last signal written, <separator> will be a  
 15365 newline character.

15366 When both the **-l** option and *exit\_status* operand are specified, the symbolic name of the  
 15367 corresponding signal will be written in the following format:

15368 "%s\n", <signal\_name>

15369 **STDERR**

15370 Used only for diagnostic messages.

15371 **OUTPUT FILES**

15372 None.

15373 **EXTENDED DESCRIPTION**

15374 None.

15375 **EXIT STATUS**

15376 The following exit values are returned:

15377 0 At least one matching process was found for each *pid* operand, and the specified signal was  
 15378 successfully processed for at least one matching process.

15379 >0 An error occurred.

15380 **CONSEQUENCES OF ERRORS**

15381 Default.

15382 **APPLICATION USAGE**

15383 Process numbers can be found by using *ps*.

15384 The job control job ID notation is not required to work as expected when *kill* is operating in its  
 15385 own utility execution environment. In either of the following examples:

```
15386 nohup kill %1 &
15387 system("kill %1");
```

15388 the *kill* operates in a different environment and will not share the shell's understanding of job  
 15389 numbers.

15390 **EXAMPLES**

15391 Any of the commands:

```
15392 kill -9 100 -165
15393 kill -s kill 100 -165
15394 kill -s KILL 100 -165
```

15395 sends the SIGKILL signal to the process whose process ID is 100 and to all processes whose  
 15396 process group ID is 165, assuming the sending process has permission to send that signal to the  
 15397 specified processes, and that they exist.

15398 The **XSH** specification and this specification do not require specific signal numbers for any  
 15399 *signal\_names*. Even the *-signal\_number* option provides symbolic (although numeric) names for  
 15400 signals. If a process is terminated by a signal, its exit status indicates the signal that killed it, but  
 15401 the exact values are not specified. The *kill -l* option, however, can be used to map decimal  
 15402 signal numbers and exit status values into the name of a signal. The following example reports  
 15403 the status of a terminated job:

```
15404 job
15405 stat=$?
15406 if [$stat -eq 0]
15407 then
15408 echo job completed successfully.
15409 elif [$stat -gt 128]
15410 then
15411 echo job terminated by signal SIG$(kill -l $stat).
15412 else
15413 echo job terminated with error code $stat.
15414 fi
```

15415 To avoid an ambiguity of an initial negative number argument specifying either a signal number  
 15416 or a process group, the ISO/IEC 9945-2: 1993 standard mandates that it always be considered the  
 15417 former. Therefore, to send the default signal to a process group (say 123), an application should  
 15418 use a command similar to one of the following:

```
15419 kill -TERM -123
15420 kill -- -123
```

15421 **FUTURE DIRECTIONS**

15422 The obsolescent forms may be withdrawn in a future issue. Applications should use the *-s*  
 15423 option.

15424 **SEE ALSO**

15425 *ps*, *wait*, the **XSH** specification description of *kill()*, *<signal.h>*.

|       |                                                 |  |
|-------|-------------------------------------------------|--|
| 15426 | <b>CHANGE HISTORY</b>                           |  |
| 15427 | First released in Issue 2.                      |  |
| 15428 | <b>Issue 4</b>                                  |  |
| 15429 | Aligned with the ISO/IEC 9945-2: 1993 standard. |  |

15430 **NAME**15431 `lex` — generate programs for lexical tasks (**DEVELOPMENT**)15432 **SYNOPSIS**15433 **OB** `lex [-c[-t]] [-n] [-v][file...]`15434 **DESCRIPTION**

15435 The *lex* utility generates C programs to be used in lexical processing of character input, and that  
 15436 can be used as an interface to *yacc*. The C programs are generated from *lex* source code and  
 15437 conform to the ISO C standard. Usually, the *lex* utility writes the program it generates to the file  
 15438 **lex.yy.c**; the state of this file is unspecified if *lex* exits with a non-zero exit status. See the  
 15439 EXTENDED DESCRIPTION section for a complete description of the *lex* input language.

15440 **OPTIONS**15441 The *lex* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

15442 The following options are supported:

15443 **OB** `-c` Indicate C-language action (default option).

15444 `-n` Suppress the summary of statistics usually written with the `-v` option. If no table sizes  
 15445 are specified in the *lex* source code and the `-v` option is not specified, then `-n` is  
 15446 implied.

15447 `-t` Write the resulting program to standard output instead of **lex.yy.c**.

15448 `-v` Write a summary of *lex* statistics to the standard output. (See the discussion of *lex* table  
 15449 sizes in **Definitions in lex** on page 425.) If the `-t` option is specified and `-n` is not  
 15450 specified, this report will be written to standard error. If table sizes are specified in the  
 15451 *lex* source code, and if the `-n` option is not specified, the `-v` option may be enabled.

15452 **OPERANDS**

15453 The following operand is supported:

15454 *file* A pathname of an input file. If more than one such *file* is specified, all files will be  
 15455 concatenated to produce a single *lex* program. If no *file* operands are specified, or if a  
 15456 *file* operand is "-", the standard input will be used.

15457 **STDIN**

15458 The standard input will be used if no *file* operands are specified, or if a *file* operand is "-". See  
 15459 **INPUT FILES**.

15460 **INPUT FILES**

15461 The input files must be text files containing *lex* source code, as described in the EXTENDED  
 15462 DESCRIPTION section.

15463 **ENVIRONMENT VARIABLES**

15464 If this variable is not set to the POSIX locale, the results are unspecified. The following  
 15465 environment variables affect the execution of *lex*:

15466 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 15467 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 15468 default locale will be used. If any of the internationalisation variables contains an  
 15469 invalid setting, the utility will behave as if none of the variables had been defined.

15470 *LC\_ALL*

15471 If set to a non-empty string value, override the values of all the other  
 15472 internationalisation variables.

15473 *LC\_COLLATE*  
 15474 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 15475 character collating elements within regular expressions. If this variable is not set to the  
 15476 POSIX locale, the results are unspecified.

15477 *LC\_CTYPE*  
 15478 Determine the locale for the interpretation of sequences of bytes of text data as  
 15479 characters (for example, single- as opposed to multi-byte characters in arguments and  
 15480 input files), and the behaviour of character classes within regular expressions. If this  
 15481 variable is not set to the POSIX locale, the results are unspecified.

15482 *LC\_MESSAGES*  
 15483 Determine the locale that should be used to affect the format and contents of diagnostic  
 15484 messages written to standard error.

15485 EX *NLSPATH*  
 15486 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

15487 **ASYNCHRONOUS EVENTS**  
 15488 Default.

15489 **STDOUT**  
 15490 If the **-t** option is specified, the text file of C source code output of *lex* will be written to standard  
 15491 output.  
 15492 If the **-t** option is not specified:

- 15493 1. Implementation-dependent informational, error and warning messages concerning the  
 15494 contents of *lex* source code input will be written to either the standard output or standard  
 15495 error.
- 15496 2. If the **-v** option is specified and the **-n** option is not specified, *lex* statistics will also be  
 15497 written to either the standard output or standard error, in an implementation-dependent  
 15498 format. These statistics may also be generated if table sizes are specified with a "%"   
 15499 operator in the *Definitions* section (see the EXTENDED DESCRIPTION section), as long as  
 15500 the **-n** option is not specified.

15501 **STDERR**  
 15502 If the **-t** option is specified, implementation-dependent informational, error and warning  
 15503 messages concerning the contents of *lex* source code input will be written to the standard error.  
 15504 If the **-t** option is not specified:

- 15505 1. Implementation-dependent informational, error and warning messages concerning the  
 15506 contents of *lex* source code input will be written to either the standard output or standard  
 15507 error.
- 15508 2. If the **-v** option is specified and the **-n** option is not specified, *lex* statistics will also be  
 15509 written to either the standard output or standard error, in an implementation-dependent  
 15510 format. These statistics may also be generated if table sizes are specified with a "%"   
 15511 operator in the *Definitions* section (see the EXTENDED DESCRIPTION section), as long as  
 15512 the **-n** option is not specified.

15513 **OUTPUT FILES**  
 15514 A text file containing C source code will be written to **lex.yy.c**, or to the standard output if the **-t**  
 15515 option is present.



15516 **EXTENDED DESCRIPTION**

15517 Each input file contains *lex* source code, which is a table of regular expressions with  
 15518 corresponding actions in the form of C program fragments.

15519 When **lex.yy.c** is compiled and linked with the *lex* library (using the **-ll** operand with *c89* or *cc*),  
 15520 the resulting program reads character input from the standard input and partitions it into strings  
 15521 that match the given expressions.

15522 When an expression is matched, these actions will occur:

- 15523 • The input string that was matched is left in *yytext* as a null-terminated string; *yytext* is either  
 15524 an external character array or a pointer to a character string. As explained in **Definitions in**  
 15525 **lex**, the type can be explicitly selected using the **%array** or **%pointer** declarations, but the  
 15526 default is implementation-dependent.
- 15527 • The external **int** *yylen* is set to the length of the matching string.
- 15528 • The expression's corresponding program fragment, or action, is executed.

15529 During pattern matching, *lex* searches the set of patterns for the single longest possible match.  
 15530 Among rules that match the same number of characters, the rule given first will be chosen.

15531 The general format of *lex* source is:

```
15532 Definitions
15533 %%
15534 Rules
15535 %%
15536 User Subroutines
```

15537 The first %% is required to mark the beginning of the rules (regular expressions and actions); the  
 15538 second %% is required only if user subroutines follow.

15539 Any line in the *Definitions* section beginning with a blank character will be assumed to be a C  
 15540 program fragment and will be copied to the external definition area of the **lex.yy.c** file.  
 15541 Similarly, anything in the *Definitions* section included between delimiter lines containing only %{  
 15542 and %} will also be copied unchanged to the external definition area of the **lex.yy.c** file.

15543 Any such input (beginning with a blank character or within %{ and %} delimiter lines) appearing  
 15544 at the beginning of the *Rules* section before any rules are specified will be written to **lex.yy.c** after  
 15545 the declarations of variables for the *yylex()* function and before the first line of code in *yylex()*.  
 15546 Thus, user variables local to *yylex()* can be declared here, as well as application code to execute  
 15547 upon entry to *yylex()*.

15548 The action taken by *lex* when encountering any input beginning with a blank character or within  
 15549 %{ and %} delimiter lines appearing in the *Rules* section but coming after one or more rules is  
 15550 undefined. The presence of such input may result in an erroneous definition of the *yylex()*  
 15551 function.

15552 **Definitions in lex**

15553 *Definitions* appear before the first %% delimiter. Any line in this section not contained between  
 15554 %{ and %} lines and not beginning with a blank character is assumed to define a *lex* substitution  
 15555 string. The format of these lines is:

```
15556 name substitute
```

If a *name* does not meet the requirements for identifiers in the ISO C standard, the result is undefined. The string *substitute* will replace the string {*name*} when it is used in a rule. The *name* string is recognised in this context only when the braces are provided and when it does not appear within a bracket expression or within double-quotes.

In the *Definitions* section, any line beginning with a "%" (percent sign) character and followed by an alphanumeric word beginning with either s or S defines a set of start conditions. Any line beginning with a "%" followed by a word beginning with either x or X defines a set of exclusive start conditions. When the generated scanner is in a %s state, patterns with no state specified will be also active; in a %x state, such patterns will not be active. The rest of the line, after the first word, is considered to be one or more blank-character-separated names of start conditions. Start condition names are constructed in the same way as definition names. Start conditions can be used to restrict the matching of regular expressions to one or more states as described in **Regular Expressions in lex** on page 427.

Implementations accept either of the following two mutually exclusive declarations in the *Definitions* section:

**%array**      Declare the type of *yytext* to be a null-terminated character array.

**%pointer**    Declare the type of *yytext* to be a pointer to a null-terminated character string.

The default type of *yytext* is implementation-dependent. If an application refers to *yytext* outside of the scanner source file (that is, via an **extern**), the application will include the appropriate **%array** or **%pointer** declaration in the scanner source file.

Implementations will accept declarations in the *Definitions* section for setting certain internal table sizes. The declarations are shown in the following table.

| Declaration        | Description                        | Minimum Value |
|--------------------|------------------------------------|---------------|
| <b>%p</b> <i>n</i> | Number of positions                | 2500          |
| <b>%n</b> <i>n</i> | Number of states                   | 500           |
| <b>%a</b> <i>n</i> | Number of transitions              | 2000          |
| <b>%e</b> <i>n</i> | Number of parse tree nodes         | 1000          |
| <b>%k</b> <i>n</i> | Number of packed character classes | 1000          |
| <b>%o</b> <i>n</i> | Size of the output array           | 3000          |

**Table 3-7** Table Size Declarations in *lex*

In the table, *n* represents a positive decimal integer, preceded by one or more blank characters. The exact meaning of these table size numbers is implementation-dependent. The implementation will document how these numbers affect the *lex* utility and how they are related to any output that may be generated by the implementation should space limitations be encountered during the execution of *lex*. It is possible to determine from this output which of the table size values needs to be modified to permit *lex* to successfully generate tables for the input language. The values in the column Minimum Value represent the lowest values conforming implementations will provide.

## 15596 Rules in lex

15597 The rules in *lex* source files are a table in which the left column contains regular expressions and  
 15598 the right column contains actions (C program fragments) to be executed when the expressions  
 15599 are recognised.

15600 *ERE action*

15601 *ERE action*

15602 ...

15603 The extended regular expression (*ERE*) portion of a row will be separated from *action* by one or  
 15604 more blank characters. A regular expression containing blank characters is recognised under  
 15605 one of the following conditions:

- 15606 • The entire expression appears within double-quotes.
- 15607 • The blank characters appear within double-quotes or square brackets.
- 15608 • Each blank character is preceded by a backslash character.

## 15609 User Subroutines in lex

15610 Anything in the user subroutines section will be copied to **lex.yy.c** following `yylex()`.

## 15611 Regular Expressions in lex

15612 The *lex* utility supports the set of extended regular expressions (see the **XBD** specification,  
 15613 **Section 7.4, Extended Regular Expressions**), with the following additions and exceptions to the  
 15614 syntax:

15615 "..." Any string enclosed in double-quotes will represent the characters within the double-  
 15616 quotes as themselves, except that backslash escapes (which appear in the following  
 15617 table) are recognised. Any backslash-escape sequence is terminated by the closing  
 15618 quote. For example, `"\01"1"` represents a single string: the octal value 1 followed by  
 15619 the character 1.

15620 `<state>r`

15621 `<state1,state2,...>r`

15622 The regular expression *r* will be matched only when the program is in one of the start  
 15623 conditions indicated by *state*, *state1* and so on; see **Actions in lex** on page 430. (As an  
 15624 exception to the typographical conventions of the rest of this specification, in this case  
 15625 `<state>` does not represent a metavariable, but the literal angle-bracket characters  
 15626 surrounding a symbol.) The start condition is recognised as such only at the beginning  
 15627 of a regular expression.

15628 *r/x* The regular expression *r* will be matched only if it is followed by an occurrence of  
 15629 regular expression *x*. The token returned in `yytext` will only match *r*. If the trailing  
 15630 portion of *r* matches the beginning of *x*, the result is unspecified. The *r* expression  
 15631 cannot include further trailing context or the `"$"` (match-end-of-line) operator; *x* cannot  
 15632 include the `"^"` (match-beginning-of-line) operator, nor trailing context, nor the `"$"`  
 15633 operator. That is, only one occurrence of trailing context is allowed in a *lex* regular  
 15634 expression, and the `"^"` operator only can be used at the beginning of such an  
 15635 expression.

**{name}** When *name* is one of the substitution symbols from the *Definitions* section, the string, including the enclosing braces, will be replaced by the *substitute* value. The *substitute* value will be treated in the extended regular expression as if it were enclosed in parentheses. No substitution will occur if **{name}** occurs within a bracket expression or within double-quotes.

Within an ERE, a backslash character is considered to begin an escape sequence as specified in the table in the **XBD** specification, **Chapter 3, File Format Notation** (`\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`). In addition, the escape sequences in the following table will be recognised.

A literal newline character cannot occur within an ERE; the escape sequence `\n` can be used to represent a newline character. A newline character cannot be matched by a period operator.

| Escape Sequence       | Description                                                                                                                                                                                                                                                                                                           | Meaning                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\digits</code>  | A backslash character followed by the longest sequence of one, two or three octal-digit characters (01234567). If all of the digits are 0, (that is, representation of the NUL character), the behaviour is undefined.                                                                                                | The character whose encoding is represented by the one-, two- or three-digit octal integer. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is implementation-dependent. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <code>\</code> for each byte. |
| <code>\xdigits</code> | A backslash character followed by the longest sequence of hexadecimal-digit characters (01234567abcdefABCDEF). If all of the digits are 0, (that is, representation of the NUL character), the behaviour is undefined.                                                                                                | The character whose encoding is represented by the hexadecimal integer.                                                                                                                                                                                                                                                                                                            |
| <code>\c</code>       | A backslash character followed by any character not described in this table or in the table in the <b>XBD</b> specification, <b>Chapter 3, File Format Notation</b> ( <code>\</code> , <code>\a</code> , <code>\b</code> , <code>\f</code> , <code>\n</code> , <code>\r</code> , <code>\t</code> , <code>\v</code> ). | The character <i>c</i> , unchanged.                                                                                                                                                                                                                                                                                                                                                |

**Table 3-8** Escape Sequences in *lex*

The order of precedence given to extended regular expressions for *lex* differs from that specified in the **XBD** specification, **Section 7.4, Extended Regular Expressions**. The order of precedence for *lex* is as shown in the following table, from high to low.

**Note:** The escaped characters entry is not meant to imply that these are operators, but they are included in the table to show their relationships to the true operators. The start condition, trailing context and anchoring notations have been omitted from the table because of the placement restrictions described in this section; they can only appear at the beginning or ending of an ERE.

| Extended Regular Expression       | Precedence           |
|-----------------------------------|----------------------|
| collation-related bracket symbols | [= =] [: :] [. .]    |
| escaped characters                | \<special character> |
| bracket expression                | []                   |
| quoting                           | "..."                |
| grouping                          | ()                   |
| definition                        | {name}               |
| single-character RE duplication   | * + ?                |
| concatenation                     |                      |
| interval expression               | {m,n}                |
| alternation                       |                      |

Table 3-9 ERE Precedence in *lex*

The ERE anchoring operators "^" and "\$" do not appear in the table. With *lex* regular expressions, these operators are restricted in their use: the "^" operator can only be used at the beginning of an entire regular expression, and the "\$" operator only at the end. The operators apply to the entire regular expression. Thus, for example, the pattern (^abc)|(def\$) is undefined; it can instead be written as two separate rules, one with the regular expression ^abc and one with def\$, which share a common action via the special "|" action (see below). If the pattern were written ^abc|def\$, it would match either **abc** or **def** on a line by itself.

Unlike the general ERE rules, embedded anchoring is not allowed by most historical *lex* implementations. An example of embedded anchoring would be for patterns such as (^|)foo(|\$) to match **foo** when it exists as a complete word. This functionality can be obtained using existing *lex* features:

```
^foo/[\n] |
" foo"/[\n] /* found foo as a separate word */
```

Note also that "\$" is a form of trailing context (it is equivalent to /\n) and as such cannot be used with regular expressions containing another instance of the operator (see the preceding discussion of trailing context).

The additional regular expressions trailing-context operator "/" can be used as an ordinary character if presented within double-quotes, "/"; preceded by a backslash, \/.; or within a bracket expression, [/]. The start-condition "<" and ">" operators are special only in a start condition at the beginning of a regular expression; elsewhere in the regular expression they are treated as ordinary characters.

The following examples clarify the differences between *lex* regular expressions and regular expressions appearing elsewhere in this specification. For regular expressions of the form *r/x*, the string matching *r* is always returned; confusion may arise when the beginning of *x* matches the trailing portion of *r*. For example, given the regular expression a\*b/cc and the input **aaabcc**, *yytext* would contain the string **aaab** on this match. But given the regular expression x\*/xy and the input **xxxxy**, the token **xxx**, not **xx**, is returned by some implementations because **xxx** matches *x\**.

In the rule ab\*/bc, the b\* at the end of *r* will extend *r*'s match into the beginning of the trailing context, so the result is unspecified. If this rule were ab/bc, however, the rule matches the text **ab** when it is followed by the text **bc**. In this latter case, the matching of *r* cannot extend into the beginning of *x*, so the result is specified.

15725      **Actions in lex**

15726      The action to be taken when an *ERE* is matched can be a C program fragment or the special  
 15727      actions described below; the program fragment can contain one or more C statements, and can  
 15728      also include special actions. The empty C statement ";" is a valid action; any string in the **lex.yy.c**  
 15729      input that matches the pattern portion of such a rule is effectively ignored or skipped. However,  
 15730      the absence of an action is not valid, and the action *lex* takes in such a condition is undefined.

15731      The specification for an action, including C statements and special actions, can extend across  
 15732      several lines if enclosed in braces:

```
15733 ERE <one or more blanks> { program statement
15734 program statement }
```

15735      The default action when a string in the input to a **lex.yy.c** program is not matched by any  
 15736      expression is to copy the string to the output. Because the default behaviour of a program  
 15737      generated by *lex* is to read the input and copy it to the output, a minimal *lex* source program that  
 15738      has just %% generates a C program that simply copies the input to the output unchanged.

15739      Four special actions are available:

```
15740 | ECHO; REJECT; BEGIN
```

15741      |      The action "|" means that the action for the next rule is the action for this rule. Unlike  
 15742      the other three actions, "|" cannot be enclosed in braces or be semicolon-terminated; it  
 15743      must be specified alone, with no other actions.

15744      ECHO;    Write the contents of the string *yytext* on the output.

15745      REJECT;

15746           Usually only a single expression is matched by a given string in the input. **REJECT**  
 15747      means "continue to the next expression that matches the current input", and causes  
 15748      whatever rule was the second choice after the current rule to be executed for the same  
 15749      input. Thus, multiple rules can be matched and executed for one input string or  
 15750      overlapping input strings. For example, given the regular expressions **xyz** and **xy** and  
 15751      the input **xyz**, usually only the regular expression **xyz** would match. The next  
 15752      attempted match would start after z. If the last action in the **xyz** rule is **REJECT**, both  
 15753      this rule and the **xy** rule would be executed. The **REJECT** action may be implemented  
 15754      in such a fashion that flow of control does not continue after it, as if it were equivalent  
 15755      to a **goto** to another part of *yylex()*. The use of **REJECT** may result in somewhat larger  
 15756      and slower scanners.

15757      BEGIN    The action:

```
15758 BEGIN newstate;
```

15759           switches the state (start condition) to *newstate*. If the string *newstate* has not been  
 15760      declared previously as a start condition in the *Definitions* section, the results are  
 15761      unspecified. The initial state is indicated by the digit 0 or the token **INITIAL**.

15762      The functions or macros described below are accessible to user code included in the *lex* input. It  
 15763      is unspecified whether they appear in the C code output of *lex*, or are accessible only through the  
 15764      **-ll** operand to *c89* or *cc* (the *lex* library).

```
15765 int yylex(void)
```

15766           Performs lexical analysis on the input; this is the primary function generated by the **lex**  
 15767      utility. The function returns zero when the end of input is reached; otherwise it returns  
 15768      non-zero values (tokens) determined by the actions that are selected.

```

15769 int yymore(void)
15770 When called, indicates that when the next input string is recognised, it is to be
15771 appended to the current value of yytext rather than replacing it; the value in yylen is
15772 adjusted accordingly.

15773 int yyless(int n)
15774 Retains n initial characters in yytext, NUL-terminated, and treats the remaining
15775 characters as if they had not been read; the value in yylen is adjusted accordingly.

15776 int input(void)
15777 Returns the next character from the input, or zero on end-of-file. It obtains input from
15778 the stream pointer yyin, although possibly via an intermediate buffer. Thus, once
15779 scanning has begun, the effect of altering the value of yyin is undefined. The character
15780 read is removed from the input stream of the scanner without any processing by the
15781 scanner.

15782 int unput(int c)
15783 Returns the character c to the input; yytext and yylen are undefined until the next
15784 expression is matched. The result of using unput for more characters than have been
15785 input is unspecified.

15786 The following functions appear only in the lex library accessible through the -ll operand; they
15787 can therefore be redefined by a portable application:

15788 int yywrap(void)
15789 Called by yylex() at end-of-file; the default yywrap() always will return 1. If the
15790 application requires yylex() to continue processing with another source of input, then
15791 the application can include a function yywrap(), which associates another file with the
15792 external variable FILE *yyin and will return a value of zero.

15793 int main(int argc, char *argv[])
15794 Calls yylex() to perform lexical analysis, then exits. The user code can contain main()
15795 to perform application-specific operations, calling yylex() as applicable.

15796 The reason for breaking these functions into two lists is that only those functions in libl.a can be
15797 reliably redefined by a portable application.

15798 Except for input(), unput() and main(), all external and static names generated by lex begin with
15799 the prefix yy or YY.

15800 EXIT STATUS
15801 The following exit values are returned:

15802 0 Successful completion.
15803 >0 An error occurred.

15804 CONSEQUENCES OF ERRORS
15805 Default.

15806 APPLICATION USAGE
15807 Portable applications are warned that in the Rules section, an ERE without an action is not
15808 acceptable, but need not be detected as erroneous by lex. This may result in compilation or run-
15809 time errors.

15810 The purpose of input() is to take characters off the input stream and discard them as far as the
15811 lexical analysis is concerned. A common use is to discard the body of a comment once the
15812 beginning of a comment is recognised.

```

The *lex* utility is not fully internationalised in its treatment of regular expressions in the *lex* source code or generated lexical analyser. It would seem desirable to have the lexical analyser interpret the regular expressions given in the *lex* source according to the environment specified when the lexical analyser is executed, but this is not possible with the current *lex* technology. Furthermore, the very nature of the lexical analysers produced by *lex* must be closely tied to the lexical requirements of the input language being described, which will frequently be locale-specific anyway. (For example, writing an analyser that is used for French text will not automatically be useful for processing other languages.)

15821 **EXAMPLES**

15822       The following is an example of a *lex* program that implements a rudimentary scanner for a  
15823       Pascal-like syntax:

```

15824 % {
15825 /* need this for the call to atof() below */
15826 #include <math.h>
15827 /* need this for printf(), fopen() and stdin below */
15828 #include <stdio.h>
15829 %}
15830 DIGIT [0-9]
15831 ID [a-z][a-z0-9]*
15832 %%
15833 {DIGIT}+ {
15834 printf("An integer: %s (%d)\n", yytext,
15835 atoi(yytext));
15836 }
15837 {DIGIT}+"."{DIGIT}* {
15838 printf("A float: %s (%g)\n", yytext,
15839 atof(yytext));
15840 }
15841 if|then|begin|end|procedure|function {
15842 printf("A keyword: %s\n", yytext);
15843 }
15844 {ID} printf("An identifier: %s\n", yytext);
15845 "+"|"-"|"*"|"|" "/" printf("An operator: %s\n", yytext);
15846 "{"|"[^]\n"}*" /* eat up one-line comments */
15847 [\t\n]+ /* eat up white space */
15848 . printf("Unrecognised character: %s\n", yytext);
15849 %%
15850 int main(int argc, char *argv[])
15851 {
15852 ++argv, --argc; /* skip over program name */
15853 if (argc > 0)
15854 yyin = fopen(argv[0], "r");
15855 else
15856 yyin = stdin;
15857 yylex();
15858 }

```

## 15859 FUTURE DIRECTIONS

15860                      None.

15861 **SEE ALSO**

15862                    *c89, yacc.*



15863 **CHANGE HISTORY**

15864 First released in Issue 2.

15865 **Issue 4**

15866 Aligned with the ISO/IEC 9945-2: 1993 standard.

15867 **NAME**15868           line — read one line (**LEGACY**)15869 **SYNOPSIS**

15870 EX       line

15871 **DESCRIPTION**

15872           The *line* utility copies one line (up to and including a newline) from the standard input and  
15873           writes it to standard output. It always writes at least a newline.

15874 **OPTIONS**

15875           None.

15876 **OPERANDS**

15877           None.

15878 **STDIN**

15879           The standard input is a text file.

15880 **INPUT FILES**

15881           None.

15882 **ENVIRONMENT VARIABLES**

15883           None.

15884 **ASYNCHRONOUS EVENTS**

15885           Default.

15886 **STDOUT**

15887           The standard output is a text file consisting of one line.

15888 **STDERR**

15889           Used only for diagnostic messages.

15890 **OUTPUT FILES**

15891           None.

15892 **EXTENDED DESCRIPTION**

15893           None.

15894 **EXIT STATUS**

15895           Exit status is:

15896           0   Successful completion.

15897           &gt;0 End-of-file on input.

15898 **CONSEQUENCES OF ERRORS**

15899           Default.

15900 **APPLICATION USAGE**15901           The *line* utility is often used within command scripts to read from the user's terminal.15902           Applications should migrate to the *read* utility.15903 **EXAMPLES**

15904           None.

15905 **FUTURE DIRECTIONS**

15906           None.

15907 **SEE ALSO**15908           *read*.

15909 **CHANGE HISTORY**

15910 First released in Issue 2.

15911 **Issue 4**

15912 Format reorganised.

15913 Marked TO BE WITHDRAWN.

15914 **Issue 5**

15915 Marked LEGACY.

15916 **NAME**15917 link — call *link()* function15918 **SYNOPSIS**15919 EX `link file1 file2`

15920

15921 **DESCRIPTION**15922 The *link* utility performs the function call:15923 `link(file1, file2);`15924 A user may need appropriate privilege to invoke the *link* utility.15925 **OPTIONS**

15926 None.

15927 **OPERANDS**

15928 The following operands are supported:

15929 *file1* The pathname of an existing file.15930 *file2* The pathname of the new directory entry to be created.15931 **STDIN**

15932 Not used.

15933 **INPUT FILES**

15934 Not used.

15935 **ENVIRONMENT VARIABLES**15936 The following environment variables affect the execution of *link*:

15937 *LANG* Provide a default value for the internationalization variables that are unset or null. If  
 15938 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 15939 default locale will be used. If any of the internationalization variables contain an  
 15940 invalid setting, the utility will behave as if none of the variables had been set.

15941 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 15942 internationalization variables.

15943 *LC\_CTYPE*

15944 Determine the locale for the interpretation of sequences of bytes of text data as  
 15945 characters (for example, single- as opposed to multi-byte characters in arguments).

15946 *LC\_MESSAGES*

15947 Determine the locale that should be used to affect the format and contents of diagnostic  
 15948 messages written to standard error.

15949 *NLSPATH*15950 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.15951 **ASYNCHRONOUS EVENTS**

15952 Default.

15953 **STDOUT**

15954 None.

15955 **STDERR**

15956 Used only for diagnostic messages.

15957 **OUTPUT FILES**

15958       None.

15959 **EXTENDED DESCRIPTION**

15960       None.

15961 **EXIT STATUS**

15962       The following exit values are returned:

15963       0   Successful completion.

15964       &gt;0  An error occurred.

15965 **CONSEQUENCES OF ERRORS**

15966       Default.

15967 **APPLICATION USAGE**

15968       None.

15969 **EXAMPLES**

15970       None.

15971 **FUTURE DIRECTIONS**

15972       None.

15973 **SEE ALSO**15974       *ln*, *unlink*, the **XSH** specification description of *link()*.15975 **CHANGE HISTORY**

15976       First released in Issue 5.

## 15977 NAME

15978 lint — check C-language programs (**DEVELOPMENT, LEGACY**)

## 15979 SYNOPSIS

```
15980 EX lint [-abcpuxv] [-D name=value]] [-I directory] [-L directory]
15981 [-o x] [-U name] operand...
```

## 15982 DESCRIPTION

15983 The *lint* utility cross-checks multiple C-language source files and library definitions and reports  
 15984 potential errors. Among the error conditions that are detected are:

- 15985 • unreachable statements
- 15986 • loops not entered at the top
- 15987 • automatic variables declared and not used
- 15988 • inconsistent declarations between files
- 15989 • non-portable constructions
- 15990 • logical expressions whose value is constant
- 15991 • functions that return values in some places and not in others
- 15992 • functions called with varying numbers or types of arguments
- 15993 • functions whose values are not used or whose values are used when none are returned.

15994 The *lint* utility takes all the files with .c and .ln suffixes, and any additional lint libraries specified  
 15995 by the **-I** operand, and processes them in their command-line order. By default, *lint* appends the  
 15996 standard C lint library to the end of the list of files. However, if the **-p** option is used, the  
 15997 portable C lint library (**lib-port.ln**), if it exists, will be appended instead. When the **-c** option is  
 15998 not used, *lint* checks this list of files for mutual compatibility. When the **-c** option is used, the .ln  
 15999 files and the lint libraries are ignored.

16000 Certain conventional comments in the C source change the behaviour of *lint*:

- |       |                                  |                                                                                |
|-------|----------------------------------|--------------------------------------------------------------------------------|
| 16001 | <code>/*NOTREACHED*/</code>      | Stop comments about unreachable code at appropriate points.                    |
| 16002 | <code>/*VARARGS<i>n</i>*/</code> | Suppress the usual checking for variable numbers of arguments in the           |
| 16003 |                                  | following function declaration. The data types of the first <i>n</i> arguments |
| 16004 |                                  | are checked; a missing <i>n</i> is taken to be zero.                           |
| 16005 | <code>/*ARGSUSED*/</code>        | Suppress diagnostic messages about unused arguments in the next                |
| 16006 |                                  | function.                                                                      |
| 16007 | <code>/*LINTLIBRARY*/</code>     | Suppress, at the beginning of a file, diagnostic messages about unused         |
| 16008 |                                  | functions and function arguments in this file. This is equivalent to using     |
| 16009 |                                  | the <b>-v</b> and <b>-x</b> options.                                           |

16010 Other comments in the C source of the form:

```
16011 /*[[[:upper:]][:upper:][:digit:]]**/
```

16012 (where the form is expressed using the syntax of basic regular expressions, defined in the **XBD**  
 16013 specification, **Section 7.3, Basic Regular Expressions**) may be interpreted by *lint* in  
 16014 implementation-dependent ways.

16015 If the **-c** option is specified, then for all pathname operands of the form *file.c* the files:

16016 `$(basename pathname .c).ln`

16017 are produced.

16018 If the **-o** option is present with option-argument *x*, a file with the name:

16019 `llib-1$(basename x).ln`

16020 is produced.

16021 The *lint* utility produces its first output on a per-source-file basis. Diagnostic messages  
 16022 regarding input files are collected and printed after all source files have been processed. Finally,  
 16023 if the **-c** option is not used, information gathered from all input files is collected and checked for  
 16024 consistency. At this point, if it is not clear whether a diagnostic message stems from a given  
 16025 source file or from one of its included files, the source filename will be printed followed by a  
 16026 question mark.

16027 During the execution of *lint*, values are established for certain predefined macros from the ISO C  
 16028 standard: `__LINE__`, `__FILE__`, `__DATE__`, `__TIME__` and `__STDC__`.

## 16029 OPTIONS

16030 The *lint* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 16031 that the **-l** operands have the format of options, but their position within a list of operands  
 16032 affects the order in which libraries are searched.

16033 The following options are supported:

16034 **-a** Suppress diagnostic messages about assignments of long values to variables that are  
 16035 not long.

16036 **-b** Suppress diagnostic messages about **break** statements that cannot be reached.

16037 **-c** Produce a **.ln** file for every **.c** file on the command line. These **.ln** files are not checked  
 16038 for interfunction compatibility.

16039 **-h** Do not apply heuristic tests that attempt to diagnose bugs intuitively, improve style  
 16040 and reduce waste.

16041 **-n** Do not check compatibility against either the standard or the portable lint library.

16042 **-o x** Produce a lint library with the name:

16043 `llib-1$(basename x).ln.`

16044 The **-c** option nullifies any use of the **-o** option. The **-o** option causes this file to be  
 16045 saved in the named lint library. To produce the lint library without extraneous  
 16046 messages, the **-x** option should be used. The **-v** option is useful if the source files for  
 16047 the lint library are just external interfaces. These option settings are also available  
 16048 through the use of the “lint comments” listed in the DESCRIPTION section.

16049 **-p** Cause all non-external names to be treated as if they were truncated to thirty-one  
 16050 characters and all external names truncated to six characters, folded to one case.  
 16051 Append the portable C lint library, if it exists, to the end of the list of files.

16052 **-u** Suppress diagnostic messages about functions and external variables used and not  
 16053 defined, or defined and not used. (This option is suitable for running *lint* on a subset of  
 16054 files of a larger program.)

16055        **-v**        Suppress diagnostic messages about unused arguments in functions.

16056        **-x**        Do not report variables referred to by external declarations but never used.

16057        The **-D**, **-U**, **-L** and **-I** options of the C compiler (see *cc* and *c89*) are also recognised as separate  
16058        arguments.

16059        The **-g** and **-O** options of the C compiler are also recognised as separate arguments, but are  
16060        ignored. (By recognising these options, the behaviour of *lint* is closer to that of the *cc* utility.)  
16061        Other options are ignored, and a warning message may be issued. The pre-defined macro *lint*  
16062        (for common-usage C) or `__LINT__` (for the ISO C standard) is defined to allow certain  
16063        questionable code to be altered or removed for *lint*.

16064        **OPERANDS**

16065        The following operands are supported:

16066        **file.c**     A pathname naming a C-language source file.

16067        **file.ln**     A pathname of a file analogous to a **.o** file produced by the C compiler.

16068        An operand of the form **-l x** means search the library named **llib-lx.ln** or **llib-lx** if **llib-lx.ln** is  
16069        not readable.

16070        The processing of other files is implementation-dependent.

16071        The *lint* utility will recognise the following **-l** operands for standard libraries:

16072        **-l c**        Names the Standard C library, **llib-lc.ln**, which will contain everything else defined by  
16073        the ISO C standard. The library searched also will include all functions defined by the  
16074        **XSH** specification. This operand is not required to be present to cause a search of the  
16075        Standard C library.

16076        **-l l**        Names the library **llib-ll.ln**, which will contain functions required by the C-language  
16077        output of *lex* that are not available through the Standard C library.

16078        **-l m**        Names the library **llib-lm.ln**, which contains the functions described by the ISO C  
16079        standard with prototypes in the **<math.h>** header.

16080        **-l pthread**  
16081        Names the library **llib-lpthread.ln** which contains the functions declared in  
16082        **<pthread.h>** and *pthread\_atfork()* referenced in **<unistd.h>**.

16083 RT        **-l rt**        Names the library **llib-lrt.ln** which contains the functions described by the Realtime  
16084        Feature Group.

16085        **-l y**        Names the library **llib-ly.ln**, which contains functions required by the C-language  
16086        output of *yacc* that are not available through the Standard C library.

16087 RT        It is unspecified whether the libraries **llib-lc.ln**, **llib-ll.ln**, **llib-lm.ln**, **llib-lpthread.ln**, **llib-lrt.ln** or  
16088        **llib-ly.ln** (or any other library accessed via **-l**) exist as regular files.

16089        **STDIN**

16090        Not used.

16091        **INPUT FILES**

16092        The input file must be one of the following: a **.c** suffixed text file containing C-language source  
16093        or a **.ln** suffixed file of unspecified format produced by a previous invocation of *lint* with a **-c** or  
16094        **-o** option.

16095        **ENVIRONMENT VARIABLES**

16096        The following environment variables may affect the execution of *lint*:



16097 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 16098 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 16099 default locale will be used. If any of the internationalisation variables contains an  
 16100 invalid setting, the utility will behave as if none of the variables had been defined.

16101 **LC\_ALL**  
 16102 If set to a non-empty string value, override the values of all the other  
 16103 internationalisation variables.

16104 **LC\_CTYPE**  
 16105 Determine the locale for the interpretation of sequences of bytes of text data as  
 16106 characters (for example, single- as opposed to multi-byte characters in arguments and  
 16107 input files).

16108 **LC\_MESSAGES**  
 16109 Determine the locale that should be used to affect the format and contents of diagnostic  
 16110 messages written to standard error.

16111 **NLSPATH**  
 16112 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

#### 16113 ASYNCHRONOUS EVENTS

16114 Default.

#### 16115 STDOUT

16116 The format of the report produced by *lint* is unspecified.

#### 16117 STDERR

16118 Used only for diagnostic and warning messages.

#### 16119 OUTPUT FILES

16120 The format of **.ln** files is unspecified.

#### 16121 EXTENDED DESCRIPTION

16122 The *lint* utility supports the same programming environments specified in *c89*, EXTENDED  
 16123 DESCRIPTION. The default programming environment supported by *lint* will be the same as  
 16124 the default programming environment supported by *c89*. To use *lint* to cross-check C-language  
 16125 source files and library definitions for other supported programming environments, provide the  
 16126 corresponding *lint* flags as the first options on the *lint* command line.

| Programming Environment<br><i>getconf</i> Name | Corresponding <i>lint</i> Arguments<br><i>getconf</i> Name |
|------------------------------------------------|------------------------------------------------------------|
| <b>_XBS5_ILP32_OFF32</b>                       | <b>XBS5_ILP32_OFF32_LINTFLAGS</b>                          |
| <b>_XBS5_ILP32_OFFBIG</b>                      | <b>XBS5_ILP32_OFFBIG_LINTFLAGS</b>                         |
| <b>_XBS5_LP64_OFF64</b>                        | <b>XBS5_LP64_OFF64_LINTFLAGS</b>                           |
| <b>_XBS5_LPBIG_OFFBIG</b>                      | <b>XBS5_LPBIG_OFFBIG_LINTFLAGS</b>                         |

16134 **Table 3-10** Programming Environments — *lint* Flags

#### 16135 EXIT STATUS

16136 The following exit values are returned:

16137 0 Successful completion.

16138 >0 An error occurred.

#### 16139 CONSEQUENCES OF ERRORS

16140 Default.

16141 **APPLICATION USAGE**

16142 The behaviour of the `-c` and the `-o` options allows for incremental use of *lint* on a set of C source  
 16143 files. Generally, *lint* is invoked once for each source file with the `-c` option. Each of these  
 16144 invocations produces a `.ln` file that corresponds to the `.c` file, and prints all messages that pertain  
 16145 to just that source file. After all the source files have been separately run through *lint*, it is  
 16146 invoked once more (without the `-c` option), listing all the `.ln` files with the needed `-l x` options.  
 16147 This will print all the interfile inconsistencies. This scheme works well with *make*; it allows *make*  
 16148 to be used to lint only the source files that have been modified since the last time the set of  
 16149 source files were checked by *lint*. If used incrementally on a set of files, the result of partial  
 16150 checks can be saved to speed subsequent intermodule consistency checks.

16151 The lint library often contains all routines in that library stripped of everything but prototypes.

16152 The intent of allowing unrecognised suffixes is to permit implementations to recognise things  
 16153 like archives and source of other languages, but not require all implementations to do so.  
 16154 Portable applications should use only the suffixes described in this specification.

16155 Programs produced by *lex* or *yacc* will often result in many diagnostic messages about **break**  
 16156 statements that cannot be reached. Use of `-b` is recommended.

16157 **EXAMPLES**

16158 Assuming a file hierarchy as shown here:

```
16159 $ ls -R
16160 libsrc:
16161 applib1.c
16162 applib2.c
16163 applib3.c
16164 applib.a
16165 appsrc:
16166 app1.c
16167 app2.c
16168 $ cd libsrc
```

16169 The following creates **llib-lapplib.ln** for later use:

```
16170 $ lint -o applib applib*.c
16171 $ cd ../appsrc
```

16172 The following checks the source for both applications against the previously created library:

```
16173 $ for app in *.c
16174 > do
16175 > lint -L ../libsrc $app -l applib
16176 > done
```

16177 If the application source and libraries in the previous example had all been built in the  
 16178 `_XBS5_LP64_OFF64` programming model, the **for** loop in the last example would be changed to:

```
16179 $ LFLAGS=$(getconf XBS5_LP64_OFF64_LINTFLAGS)
16180 $ for app in *.c
16181 > do
16182 > lint $LFLAGS -L ../libsrc $app -l applib
16183 > done
```

16184 **FUTURE DIRECTIONS**

16185 None.

16186 **SEE ALSO**16187           *cc, c89, getconf, lex, make, yacc.* |16188 **CHANGE HISTORY**

16189           First released in Issue 2. |

16190 **Issue 4**

16191           Format reorganised.

16192           Exceptions to Utility Syntax Guidelines conformance noted.

16193           Internationalised environment variable support made optional. |

16194 **Issue 5** |

16195           Marked LEGACY.

## 16196 NAME

16197 ln — link files

## 16198 SYNOPSIS

16199 ln [-f] *source\_file target\_file*16200 ln [-f] *source\_file... target\_dir*

## 16201 DESCRIPTION

16202 In the first synopsis form, the *ln* utility will create a new directory entry (link) for the file  
 16203 specified by the *source\_file* operand, at the *destination* path specified by the *target\_file* operand.  
 16204 This first synopsis form is assumed when the final operand does not name an existing directory;  
 16205 if more than two operands are specified and the final is not an existing directory, an error will  
 16206 result.

16207 In the second synopsis form, the *ln* utility will create a new directory entry for each file specified  
 16208 by a *source\_file* operand, at a *destination* path in the existing directory named by *target\_dir*.

16209 If the last operand specifies an existing file of a type not specified by the **XSH** specification, the  
 16210 behaviour is implementation-dependent.

16211 The corresponding destination path for each *source\_file* will be the concatenation of the target  
 16212 directory pathname, a slash character, and the last pathname component of the *source\_file*. The  
 16213 second synopsis form will be assumed when the final operand names an existing directory.

16214 For each *source\_file*:

- 16215 1. If the *destination* path exists:
  - 16216 a. If the **-f** option is not specified, *ln* will write a diagnostic message to standard error,  
 16217 do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.
  - 16218 b. Actions will be performed equivalent to the **XSH** specification *unlink()* function,  
 16219 called using *destination* as the *path* argument. If this fails for any reason, *ln* will write  
 16220 a diagnostic message to standard error, do nothing more with the current *source\_file*,  
 16221 and go on to any remaining *source\_files*.
- 16222 2. Actions will be performed equivalent to the **XSH** specification *link()* function using  
 16223 *source\_file* as the *path1* argument, and the *destination* path as the *path2* argument.

## 16224 OPTIONS

16225 The *ln* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

16226 The following option is supported:

16227 **-f** Force existing *destination* pathnames to be removed to allow the link.

## 16228 OPERANDS

16229 The following operands are supported:

16230 *source\_file*

16231 A pathname of a file to be linked. This can be a regular or special file; whether a  
 16232 directory can be linked is implementation-dependent.

16233 *target\_file*

16234 The pathname of the new directory entry to be created.

16235 *target\_dir*

16236 A pathname of an existing directory in which the new directory entries are to be  
 16237 created.

16238 **STDIN**

16239 Not used.

16240 **INPUT FILES**

16241 None.

16242 **ENVIRONMENT VARIABLES**16243 The following environment variables affect the execution of *ln*:

16244 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 16245 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 16246 default locale will be used. If any of the internationalisation variables contains an  
 16247 invalid setting, the utility will behave as if none of the variables had been defined.

16248 *LC\_ALL*

16249 If set to a non-empty string value, override the values of all the other  
 16250 internationalisation variables.

16251 *LC\_CTYPE*

16252 Determine the locale for the interpretation of sequences of bytes of text data as  
 16253 characters (for example, single- as opposed to multi-byte characters in arguments).

16254 *LC\_MESSAGES*

16255 Determine the locale that should be used to affect the format and contents of diagnostic  
 16256 messages written to standard error.

16257 EX *NLSPATH*16258 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.16259 **ASYNCHRONOUS EVENTS**

16260 Default.

16261 **STDOUT**

16262 Not used.

16263 **STDERR**

16264 Used only for diagnostic messages.

16265 **OUTPUT FILES**

16266 None.

16267 **EXTENDED DESCRIPTION**

16268 None.

16269 **EXIT STATUS**

16270 The following exit values are returned:

16271 0 All the specified files were linked successfully.

16272 &gt;0 An error occurred.

16273 **CONSEQUENCES OF ERRORS**

16274 Default.

16275 **APPLICATION USAGE**

16276 None.

16277 **EXAMPLES**

16278 None.

16279 **FUTURE DIRECTIONS**

16280 None.

16281 **SEE ALSO**16282 *chmod, find, pax, rm.*16283 **CHANGE HISTORY**

16284 First released in Issue 2.

16285 **Issue 4**

16286 Aligned with the ISO/IEC 9945-2: 1993 standard.

16287 **NAME**

16288       locale — get locale-specific information

16289 **SYNOPSIS**

16290       locale [ -a | -m ]

16291       locale [-ck] *name*...16292 **DESCRIPTION**

16293       The *locale* utility writes information about the current locale environment, or all public locales, to  
 16294       the standard output. For the purposes of this section, a *public locale* is one provided by the  
 16295       implementation that is accessible to the application.

16296       When *locale* is invoked without any arguments, it summarises the current locale environment for  
 16297       each locale category as determined by the settings of the environment variables defined in the  
 16298       **XBD specification, Chapter 5, Locale**.

16299       When invoked with operands, it writes values that have been assigned to the keywords in the  
 16300       locale categories, as follows:

- 16301       • Specifying a keyword name selects the named keyword and the category containing that  
 16302       keyword.
- 16303       • Specifying a category name selects the named category and all keywords in that category.

16304 **OPTIONS**16305       The *locale* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**.

16306       The following options are supported:

- 16307       **-a**       Write information about all available public locales. The available locales include  
 16308       **POSIX**, representing the POSIX locale. The manner in which the implementation  
 16309       determines what other locales are available is implementation-dependent.
- 16310       **-c**       Write the names of selected locale categories; see the STDOUT section. The **-c** option  
 16311       increases readability when more than one category is selected (for example, via more  
 16312       than one keyword name or via a category name). It is valid both with and without the  
 16313       **-k** option.
- 16314       **-k**       Write the names and values of selected keywords. The implementation may omit  
 16315       values for some keywords; see the OPERANDS section.
- 16316       **-m**       Write names of available charmaps; see the **XBD specification, Section 4.1, Portable**  
 16317       **Character Set**.

16318 **OPERANDS**

16319       The following operand is supported:

- 16320       *name*     The name of a locale category as defined in the **XBD specification, Chapter 5, Locale**,  
 16321       the name of a keyword in a locale category, or the reserved name **charmap**. The named  
 16322       category or keyword will be selected for output. If a single *name* represents both a  
 16323       locale category name and a keyword name in the current locale, the results are  
 16324       unspecified. Otherwise, both category and keyword names can be specified as *name*  
 16325       operands, in any sequence. It is implementation-dependent whether any keyword  
 16326       values are written for the categories LC\_CTYPE and LC\_COLLATE.

16327 **STDIN**

16328       Not used.

16329 **INPUT FILES**

16330       None.

16331 **ENVIRONMENT VARIABLES**16332       The following environment variables affect the execution of *locale*:

16333       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 16334       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 16335       default locale will be used. If any of the internationalisation variables contains an  
 16336       invalid setting, the utility will behave as if none of the variables had been defined.

16337       **LC\_ALL**

16338       If set to a non-empty string value, override the values of all the other  
 16339       internationalisation variables.

16340       **LC\_CTYPE**

16341       Determine the locale for the interpretation of sequences of bytes of text data as  
 16342       characters (for example, single- as opposed to multi-byte characters in arguments and  
 16343       input files).

16344       **LC\_MESSAGES**

16345       Determine the locale that should be used to affect the format and contents of diagnostic  
 16346       messages written to standard error.

16347 EX       **NLSPATH**16348       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

16349 EX       The *LANG*, *LC\_\** and *NLSPATH* environment variables must specify the current locale  
 16350       environment to be written out; they will be used if the **-a** option is not specified.

16351 **ASYNCHRONOUS EVENTS**

16352       Default.

16353 **STDOUT**

16354       If *locale* is invoked without any options or operands, the names and values of the *LANG* and  
 16355       *LC\_\** environment variables described in this specification will be written to the standard  
 16356       output, one variable per line, with *LANG* first, and each line using the following format. Only  
 16357       those variables set in the environment and not overridden by *LC\_ALL* will be written using this  
 16358       format:

16359       "*%s=%s\n*", *<variable\_name>*, *<value>*

16360       The names of those *LC\_\** variables associated with locale categories defined in this specification  
 16361       that are not set in the environment or are overridden by *LC\_ALL* will be written in the following  
 16362       format:

16363       "*%s=\"%s\"\\n*", *<variable\_name>*, *<implied value>*

16364       The *<implied value>* is the name of the locale that has been selected for that category by the  
 16365       implementation, based on the values in *LANG* and *LC\_ALL*, as described in the **XBD**  
 16366       specification, **Chapter 6, Environment Variables**.

16367       The *<value>* and *<implied value>* shown above will be properly quoted for possible later reentry  
 16368       to the shell. The *<value>* will not be quoted using double-quotes (so that it can be distinguished  
 16369       by the user from the *<implied value>* case, which always requires double-quotes).



16370 The *LC\_ALL* variable will be written last, using the first format shown above. If it is not set, it  
 16371 will be written as:

16372 "LC\_ALL=\n"

16373 If any arguments are specified:

- 16374 1. If the **-a** option is specified, the names of all the public locales will be written, each in the  
 16375 following format:

16376 "%s\n", <locale name>

- 16377 2. If the **-c** option is specified, the names of all selected categories will be written, each in the  
 16378 following format:

16379 "%s\n", <category name>

16380 If keywords are also selected for writing (see following items), the category name output  
 16381 will precede the keyword output for that category.

16382 If the **-c** option is not specified, the names of the categories will not be written; only the  
 16383 keywords, as selected by the *name* operand, will be written.

- 16384 3. If the **-k** option is specified, the names and values of selected keywords will be written. If  
 16385 a value is non-numeric, it will be written in the following format:

16386 "%s=\"%s\"\\n\", <keyword name>, <keyword value>

16387 If the keyword was **charmap**, the name of the charmap (if any) that was specified via the  
 16388 *localedef* **-f** option when the locale was created will be written, with the word **charmap** as  
 16389 <keyword name>.

16390 If a value is numeric, it will be written in one of the following formats:

16391 "%s=%d\\n\", <keyword name>, <keyword value>

16392 "%s=%c%o\\n\", <keyword name>, <escape character>, <keyword value>

16393 "%s=%cx%x\\n\", <keyword name>, <escape character>,  
 16394 <keyword value>

16395 where the <escape character> is that identified by the **escape\_char** keyword in the current  
 16396 locale; see the XBD specification, **Section 5.3, Locale Definition**.

16397 Compound keyword values (list entries) will be separated in the output by semicolons.  
 16398 When included in keyword values, the semicolon, the double-quote, the backslash and any  
 16399 control character will be preceded (escaped) with the escape character.

- 16400 4. If the **-k** option is not specified, selected keyword values will be written, each in the  
 16401 following format:

16402 "%s\\n\", <keyword value>

16403 If the keyword was **charmap**, the name of the charmap (if any) that was specified via the  
 16404 *localedef* **-f** option when the locale was created will be written.

- 16405 5. If the **-m** option is specified, then a list of all available charmaps will be written, each in  
 16406 the format:

16407 "%s\\n\", <charmap>

16408 where <charmap> is in a format suitable for use as the option-argument to the *localedef* **-f**  
 16409 option.

16410 **STDERR**

16411       Used only for diagnostic messages.

16412 **OUTPUT FILES**

16413       None.

16414 **EXTENDED DESCRIPTION**

16415       None.

16416 **EXIT STATUS**

16417       The following exit values are returned:

16418       0   All the requested information was found and output successfully.

16419       &gt;0   An error occurred.

16420 **CONSEQUENCES OF ERRORS**

16421       Default.

16422 **APPLICATION USAGE**

16423       If the *LANG* environment variable is not set or set to an empty value, or one of the *LC\_\**  
 16424       environment variables is set to an unrecognised value, the actual locales assumed (if any) are  
 16425       implementation-dependent as described in the **XBD** specification, **Chapter 6, Environment**  
 16426       **Variables**.

16427       Implementations are not required to write out the actual values for keywords in the categories  
 16428       *LC\_CTYPE* and *LC\_COLLATE*; however, they must write out the categories (allowing an  
 16429       application to determine, for example, which character classes are available).

16430 **EXAMPLES**

16431       In the following examples, the assumption is that locale environment variables are set as  
 16432       follows:

16433       LANG=locale\_x

16434       LC\_COLLATE=locale\_y

16435       The command:

16436       locale

16437       would result in the following output:

16438       LANG=locale\_x

16439       LC\_CTYPE="locale\_x"

16440       LC\_COLLATE=locale\_y

16441       LC\_TIME="locale\_x"

16442       LC\_NUMERIC="locale\_x"

16443       LC\_MONETARY="locale\_x"

16444       LC\_MESSAGES="locale\_x"

16445       LC\_ALL=

16446       The order of presentation of the categories is not specified by this specification. |

16447       The command:

16448       LC\_ALL=POSIX locale -ck decimal\_point

16449       would produce:

16450       LC\_NUMERIC

16451       decimal\_point="."

16452       The following command shows an application of *locale* to determine whether a user-supplied  
16453       response is affirmative:

```
16454 if printf "%s\n" "$response" | grep -Eq "$(locale yesexpr)"
16455 then
16456 affirmative processing goes here
16457 else
16458 non-affirmative processing goes here
16459 fi
```

#### 16460 **FUTURE DIRECTIONS**

16461       The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
16462       interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
16463       corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
16464       finalised.

#### 16465 **SEE ALSO**

16466       *localedef*, the **XBD** specification, **Section 5.3, Locale Definition**.

#### 16467 **CHANGE HISTORY**

16468       First released in Issue 4.

#### 16469 **Issue 5**

16470       FUTURE DIRECTIONS section added.

16471 **NAME**

16472 localedef — define locale environment

16473 **SYNOPSIS**

16474 localedef [-c][-f *charmap*][-i *sourcefile*] *name*

16475 **DESCRIPTION**

16476 The *localedef* utility converts source definitions for locale categories into a format usable by the  
 16477 functions and utilities whose operational behaviour is determined by the setting of the locale  
 16478 environment variables defined in the **XBD** specification, **Chapter 5, Locale**. It is  
 16479 implementation-dependent whether users have the capability to create new locales, in addition  
 16480 EX to those supplied by the implementation. Since the symbolic constant {POSIX2\_LOCALEDEF} is  
 16481 defined on all XSI-conformant systems, the system supports the creation of new locales.

16482 The utility reads source definitions for one or more locale categories belonging to the same  
 16483 locale from the file named in the *-i* option (if specified) or from standard input.

16484 The *name* operand identifies the target locale. The utility supports the creation of *public*, or  
 16485 generally accessible locales, as well as *private*, or restricted-access locales. Implementations may  
 16486 restrict the capability to create or modify public locales to users with the appropriate privileges.

16487 Each category source definition is identified by the corresponding environment variable name  
 16488 and terminated by an **END** *category-name* statement. The following categories are supported. In  
 16489 addition, the input may contain source for implementation-dependent categories.

16490 LC\_CTYPE

16491 Defines character classification and case conversion.

16492 LC\_COLLATE

16493 Defines collation rules.

16494 LC\_MONETARY

16495 Defines the format and symbols used in formatting of monetary information.

16496 LC\_NUMERIC

16497 Defines the decimal delimiter, grouping and grouping symbol for non-monetary  
 16498 numeric editing.

16499 LC\_TIME

16500 Defines the format and content of date and time information.

16501 LC\_MESSAGES

16502 Defines the format and values of affirmative and negative responses.

16503 **OPTIONS**

16504 The *localedef* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

16505 The following options are supported:

16506 *-c* Create permanent output even if warning messages have been issued.

16507 *-f charmap*

16508 Specify the pathname of a file containing a mapping of character symbols and collating  
 16509 element symbols to actual character encodings. The format of the *charmap* is described  
 16510 under the **XBD** specification, **Section 4.4, Character Set Description File**. This option  
 16511 must be specified if symbolic names (other than collating symbols defined in a  
 16512 **collating-symbol** keyword) are used. If the *-f* option is not present, an  
 16513 implementation-dependent character mapping will be used.

16514        -i *inputfile*  
 16515            The pathname of a file containing the source definitions. If this option is not present,  
 16516            source definitions will be read from standard input. The format of the *inputfile* is  
 16517            described in the **XBD** specification, **Section 5.3, Locale Definition**.

16518 **OPERANDS**  
 16519        The following operand is supported:

16520        *name*     Identifies the locale. See the **XBD** specification, **Chapter 5, Locale** for a description of  
 16521            the use of this name. If the name contains one or more slash characters, *name* will be  
 16522            interpreted as a pathname where the created locale definitions will be stored. If *name*  
 16523            does not contain any slash characters, the interpretation of the name is  
 16524            implementation-dependent and the locale will be public. This capability may be  
 16525            restricted to users with appropriate privileges. (As a consequence of specifying one  
 16526            *name*, although several categories can be processed in one execution, only categories  
 16527            belonging to the same locale can be processed.)

16528 **STDIN**  
 16529        Unless the -i option is specified, the standard input must be a text file containing one or more  
 16530        locale category source definitions, as described in the **XBD** specification, **Section 5.3, Locale**  
 16531        **Definition**. When lines are continued using the escape character mechanism, there is no limit to  
 16532        the length of the accumulated continued line.

16533 **INPUT FILES**  
 16534        The character set mapping file specified as the *charmap* option-argument is described under the  
 16535        **XBD** specification, **Section 4.4, Character Set Description File**. If a locale category source  
 16536        definition contains a **copy** statement, as defined in the **XBD** specification, **Chapter 5, Locale**, and  
 16537        the **copy** statement names a valid, existing locale, then *localedef* will behave as if the source  
 16538        definition had contained a valid category source definition for the named locale.

16539 **ENVIRONMENT VARIABLES**  
 16540        The following environment variables affect the execution of *localedef*:

16541        *LANG*     Provide a default value for the internationalisation variables that are unset or null. If  
 16542            *LANG* is unset or null, the corresponding value from the implementation-dependent  
 16543            default locale will be used. If any of the internationalisation variables contains an  
 16544            invalid setting, the utility will behave as if none of the variables had been defined.

16545        *LC\_ALL*  
 16546            If set to a non-empty string value, override the values of all the other  
 16547            internationalisation variables.

16548        *LC\_COLLATE*  
 16549            (This variable has no affect on *localedef*; the POSIX locale will be used for this category.)

16550        *LC\_CTYPE*  
 16551            Determine the locale for the interpretation of sequences of bytes of text data as  
 16552            characters (for example, single- as opposed to multi-byte characters in arguments and  
 16553            input files). This variable has no affect on the processing of *localedef* input data; the  
 16554            POSIX locale is used for this purpose, regardless of the value of this variable.

16555        *LC\_MESSAGES*  
 16556            Determine the locale that should be used to affect the format and contents of diagnostic  
 16557            messages written to standard error.

16558 EX        *NLSPATH*  
 16559            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

16560 **ASYNCHRONOUS EVENTS**

16561 Default.

16562 **STDOUT**

16563 The utility will report all categories successfully processed, in an unspecified format.

16564 **STDERR**

16565 Used only for diagnostic messages.

16566 **OUTPUT FILES**

16567 The format of the created output is unspecified. If the *name* operand does not contain a slash,  
16568 the existence of an output file for the locale is unspecified.

16569 **EXTENDED DESCRIPTION**

16570 None.

16571 **EXIT STATUS**

16572 The following exit values are returned:

- 16573 0 No errors occurred and the locales were successfully created.  
16574 1 Warnings occurred and the locales were successfully created.  
16575 2 The locale specification exceeded implementation limits or the coded character set or sets  
16576 used were not supported by the implementation, and no locale was created.  
16577 3 The capability to create new locales is not supported by the implementation.  
16578 >3 Warnings or errors occurred and no output was created.

16579 **CONSEQUENCES OF ERRORS**

16580 If an error is detected, no permanent output will be created.

16581 If warnings occur, permanent output will be created if the **-c** option was specified. The  
16582 following conditions will cause warning messages to be issued:

- 16583 • If a symbolic name not found in the *charmap* file is used for the descriptions of the LC\_CTYPE  
16584 or LC\_COLLATE categories (for other categories, this will be an error condition).
- 16585 • If the number of operands to the **order** keyword exceeds the {COLL\_WEIGHTS\_MAX} limit.
- 16586 • If optional keywords not supported by the implementation are present in the source.

16587 Other implementation-dependent conditions may also cause warnings.

16588 **APPLICATION USAGE**

16589 The *charmap* definition is optional, and is contained outside the locale definition. This allows  
16590 both completely self-defined source files, and generic sources (applicable to more than one  
16591 codeset). To aid portability, all *charmap* definitions must use the same symbolic names for the  
16592 portable character set. As explained in the **XBD** specification, **Section 4.4, Character Set  
16593 Description File**, it is implementation-dependent whether or not users or applications can  
16594 provide additional character set description files. Therefore, the **-f** option might be operable  
16595 only when an implementation-dependent *charmap* is named.

16596 **EXAMPLES**

16597 None.

16598 **FUTURE DIRECTIONS**

16599 None.

16600 **SEE ALSO**16601 *locale*, the **XBD** specification, **Section 5.3, Locale Definition**.

16602 **CHANGE HISTORY**

16603           First released in Issue 4.

16604 **NAME**

16605       logger — log messages

16606 **SYNOPSIS**16607       logger *string* ...16608 **DESCRIPTION**

16609       The *logger* utility saves a message, in an unspecified manner and format, containing the *string*  
 16610       operands provided by the user. The messages are expected to be evaluated later by personnel  
 16611       performing system administration tasks.

16612       It is implementation-dependent whether messages written in locales other than the POSIX locale  
 16613       are effective.

16614 **OPTIONS**

16615       None.

16616 **OPERANDS**

16617       The following operand is supported:

16618       *string*   One of the string arguments whose contents are concatenated together, in the order  
 16619       specified, separated by single space characters.

16620 **STDIN**

16621       Not used.

16622 **INPUT FILES**

16623       None.

16624 **ENVIRONMENT VARIABLES**16625       The following environment variables affect the execution of *logger*:

16626       *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 16627       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 16628       default locale will be used. If any of the internationalisation variables contains an  
 16629       invalid setting, the utility will behave as if none of the variables had been defined.

16630       *LC\_ALL*

16631       If set to a non-empty string value, override the values of all the other  
 16632       internationalisation variables.

16633       *LC\_CTYPE*

16634       Determine the locale for the interpretation of sequences of bytes of text data as  
 16635       characters (for example, single- as opposed to multi-byte characters in arguments).

16636       *LC\_MESSAGES*

16637       Determine the locale that should be used to affect the format and contents of diagnostic  
 16638       messages written to standard error. (This means diagnostics from *logger* to the user or  
 16639       application, not diagnostic messages that the user is sending to the system  
 16640       administrator.)

16641 EX       *NLSPATH*16642       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.16643 **ASYNCHRONOUS EVENTS**

16644       Default.

16645 **STDOUT**

16646       Not used.



**16647 STDERR**

16648           Used only for diagnostic messages.

**16649 OUTPUT FILES**

16650           Unspecified.

**16651 EXTENDED DESCRIPTION**

16652           None.

**16653 EXIT STATUS**

16654           The following exit values are returned:

16655           0   Successful completion.

16656           >0  An error occurred.

**16657 CONSEQUENCES OF ERRORS**

16658           Default.

**16659 APPLICATION USAGE**

16660           This utility allows logging of information for later use by a system administrator or programmer  
16661           in determining why non-interactive utilities have failed. The locations of the saved messages,  
16662           their format and retention period are all unspecified. There is no method for a portable  
16663           application to read messages, once written.

**16664 EXAMPLES**

16665           A batch application, running non-interactively, tries to read a configuration file and fails; it may  
16666           attempt to notify the system administrator with:

16667           logger myname: unable to read file foo. [timestamp]

**16668 FUTURE DIRECTIONS**

16669           None.

**16670 SEE ALSO**

16671           *mailx*, *write*.

**16672 CHANGE HISTORY**

16673           First released in Issue 4.

16674 **NAME**

16675 logname — return the user's login name

16676 **SYNOPSIS**

16677 logname

16678 **DESCRIPTION**

16679 The *logname* utility will write the user's login name to standard output. The login name is the  
 16680 string that would be returned by the **XSH** specification *getlogin()* function. Under the conditions  
 16681 where the *getlogin()* function would fail, the *logname* utility will write a diagnostic message to  
 16682 standard error and exit with a non-zero exit status.

16683 **OPTIONS**

16684 None.

16685 **OPERANDS**

16686 None.

16687 **STDIN**

16688 Not used.

16689 **INPUT FILES**

16690 None.

16691 **ENVIRONMENT VARIABLES**16692 The following environment variables affect the execution of *logname*:

16693 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 16694 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 16695 default locale will be used. If any of the internationalisation variables contains an  
 16696 invalid setting, the utility will behave as if none of the variables had been defined.

16697 **LC\_ALL**

16698 If set to a non-empty string value, override the values of all the other  
 16699 internationalisation variables.

16700 **LC\_CTYPE**

16701 Determine the locale for the interpretation of sequences of bytes of text data as  
 16702 characters (for example, single- as opposed to multi-byte characters in arguments).

16703 **LC\_MESSAGES**

16704 Determine the locale that should be used to affect the format and contents of diagnostic  
 16705 messages written to standard error.

16706 EX **NLSPATH**16707 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.16708 **ASYNCHRONOUS EVENTS**

16709 Default.

16710 **STDOUT**16711 The *logname* utility output will be a single line consisting of the user's login name:

16712 "%s\n", &lt;login name&gt;

16713 **STDERR**

16714 Used only for diagnostic messages.

16715 **OUTPUT FILES**

16716 None.

16717 **EXTENDED DESCRIPTION**

16718           None.

16719 **EXIT STATUS**

16720           The following exit values are returned:

16721           0   Successful completion.

16722           &gt;0  An error occurred.

16723 **CONSEQUENCES OF ERRORS**

16724           Default.

16725 **APPLICATION USAGE**16726           The *logname* utility explicitly ignores the *LOGNAME* environment variable because environment changes could produce erroneous results.16728 **EXAMPLES**

16729           None.

16730 **FUTURE DIRECTIONS**

16731           None.

16732 **SEE ALSO**16733           *id*, *who*.16734 **CHANGE HISTORY**

16735           First released in Issue 2. |

16736 **Issue 4**

16737           Aligned with the ISO/IEC 9945-2: 1993 standard.

## 16738 NAME

16739 lp — send files to a printer

## 16740 SYNOPSIS

16741 lp [-c][-d *dest*][-n *copies*][-msw][-o *option*]... [-t *title*]  
 16742 [*file*...]

## 16743 DESCRIPTION

16744 The *lp* utility copies the input files to an output destination in an unspecified manner. The  
 16745 default output destination should be to a hardcopy device, such as a printer or microfilm  
 16746 recorder, that produces non-volatile, human-readable documents. If such a device is not  
 16747 available to the application, or if the system provides no such device, the *lp* utility will exit with  
 16748 a non-zero exit status.

16749 The actual writing to the output device may occur some time after the *lp* utility successfully  
 16750 exits. During the portion of the writing that corresponds to each input file, the implementation  
 16751 guarantees exclusive access to the device.

16752 EX The *lp* utility associates a unique *request ID* with each request.

16753 Normally, a banner page is produced to separate and identify each print job. This page may be  
 16754 suppressed by implementation-dependent conditions, such as an operator command or one of  
 16755 the **-o** *option* values.

## 16756 OPTIONS

16757 The *lp* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

16758 The following options are supported:

16759 **-c** Exit only after further access to any of the input files is no longer required. The  
 16760 application can then safely delete or modify the files without affecting the output  
 16761 operation. Normally, files will not be copied, but will be linked whenever possible. If  
 16762 the **-c** option is not given, then the user should be careful not to remove any of the files  
 16763 before the request has been printed in its entirety. It should also be noted that in the  
 16764 absence of the **-c** option, any changes made to the named files after the request is made  
 16765 but before it is printed will be reflected in the printed output. On some systems, **-c**  
 16766 may be on by default.

16767 **-d** *dest*

16768 EX Specify a string that names the destination (*dest*). If *dest* is a printer, the request will be  
 16769 printed only on that specific printer. If *dest* is a class of printers, the request will be  
 16770 printed on the first available printer that is a member of the class. Under certain  
 16771 conditions (printer unavailability, file space limitation, and so on), requests for specific  
 16772 destinations need not be accepted; see *lpstat*. Destination names vary between systems;  
 16773 see *lpstat*.

16774 If **-d** is not specified, and neither the *LPDEST* nor *PRINTER* environment variable is set,  
 16775 an unspecified destination is used. The **-d** *dest* option takes precedence over *LPDEST*,  
 16776 which in turn takes precedence over *PRINTER*. Results are undefined when *dest*  
 16777 contains a value that is not a valid destination name.

16778 EX UN **-m** Send mail (see *mailx*) after the files have been printed. By default, no mail is sent upon  
 16779 normal completion of the print request.

16780 **-n** *copies*

16781 Write *copies* number of copies of the files, where *copies* is a positive decimal integer.  
 16782 The methods for producing multiple copies and for arranging the multiple copies when  
 16783 multiple *file* operands are used are unspecified, except that each file will be output as

16784 an integral whole, not interleaved with portions of other files.

16785 EX UN **-o option**  
 16786 Specify printer-dependent or class-dependent *options*. Several such *options* may be  
 16787 collected by specifying the **-o** option more than once.

16788 EX PI **-s** Suppress messages from *lp* such as “request id is ...”.

16789 EX UN **-t title** Write *title* on the banner page of the output.

16790 EX UN **-w** Write a message on the user’s terminal after the files have been printed. If the user is  
 16791 not logged in, then mail will be sent instead.

## 16792 OPERANDS

16793 The following operand is supported:

16794 **file** A pathname of a file to be output. If no *file* operands are specified, or if a *file* operand is  
 16795 “-”, the standard input will be used. If a *file* operand is used, but the **-c** option is not  
 16796 specified, the process performing the writing to the output device may have user and  
 16797 group permissions that differ from that of the process invoking *lp*.

## 16798 STDIN

16799 The standard input will be used only if no *file* operands are specified, or if a *file* operand is “-”.  
 16800 See the INPUT FILES section.

## 16801 INPUT FILES

16802 The input files must be text files.

## 16803 ENVIRONMENT VARIABLES

16804 The following environment variables affect the execution of *lp*:

16805 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 16806 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 16807 default locale will be used. If any of the internationalisation variables contains an  
 16808 invalid setting, the utility will behave as if none of the variables had been defined.

16809 **LC\_ALL**  
 16810 If set to a non-empty string value, override the values of all the other  
 16811 internationalisation variables.

16812 **LC\_CTYPE**  
 16813 Determine the locale for the interpretation of sequences of bytes of text data as  
 16814 characters (for example, single- as opposed to multi-byte characters in arguments and  
 16815 input files).

16816 **LC\_MESSAGES**  
 16817 Determine the locale that should be used to affect the format and contents of diagnostic  
 16818 messages written to standard error and informative messages written to standard  
 16819 output.

16820 EX **LC\_TIME**  
 16821 Determine the format and contents of date and time strings displayed in the *lp* banner  
 16822 page, if any.

16823 **LPDEST**  
 16824 Determine the destination. If the *LPDEST* environment variable is not set, the  
 16825 *PRINTER* environment variable will be used. The **-d dest** option takes precedence  
 16826 over *LPDEST*. Results are undefined when **-d** is not specified and *LPDEST* contains a  
 16827 value that is not a valid destination name.

16828 EX **NLSPATH**  
 16829 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

16830 **PRINTER**  
 16831 Determine the output device or destination. If the *LPDEST* and *PRINTER* environment  
 16832 variables are not set, an unspecified output device is used. The *-d dest* option and the  
 16833 *LPDEST* environment variable takes precedence over *PRINTER*. Results are undefined  
 16834 when *-d* is not specified, *LPDEST* is unset, and *PRINTER* contains a value that is not a  
 16835 valid device or destination name.

16836 **ASYNCHRONOUS EVENTS**  
 16837 Default.

16838 **STDOUT**  
 16839 EX The *lp* utility writes a *request ID* to the standard output, unless *-s* is specified. The format of the  
 16840 message is unspecified. This *request ID* can be used later to cancel (see *cancel*) or find the status  
 16841 (see *lpstat*) of the request.

16842 **STDERR**  
 16843 Used only for diagnostic messages.

16844 **OUTPUT FILES**  
 16845 None.

16846 **EXTENDED DESCRIPTION**  
 16847 None.

16848 **EXIT STATUS**  
 16849 The following exit values are returned:  
 16850 0 All input files were processed successfully.  
 16851 >0 No output device was available, or an error occurred.

16852 **CONSEQUENCES OF ERRORS**  
 16853 Default.

16854 **APPLICATION USAGE**  
 16855 The *pr* and *fold* utilities can be used to achieve reasonable formatting for the implementation's  
 16856 default page size.  
 16857 A portable application can use one of the *file* operands only with the *-c* option or if the file is  
 16858 publicly readable and guaranteed to be available at the time of printing. This is because the  
 16859 standard gives the implementation the freedom to queue up the request for printing at some  
 16860 later time by a different process that might not be able to access the file.

16861 **EXAMPLES**  
 16862 1. To print file *file*:  
 16863 `lp -c file`  
 16864 2. To print multiple files with headers:  
 16865 `pr file1 file2 | lp`

16866 **FUTURE DIRECTIONS**  
 16867 None.

16868 **SEE ALSO**  
 16869 *banner*, *lpstat*, *mailx*.

16870 **CHANGE HISTORY**

16871 First released in Issue 2.

16872 **Issue 4**

16873 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 16874 NAME

16875 lpstat — report printer status information (**LEGACY**)

## 16876 SYNOPSIS

16877 EX lpstat [-drst] [-a[*list*]] [-c[*list*]] [-o[*list*]] [-p[*list*]] [-u[*list*]]  
 16878 [-v[*list*]] [*ID*...]

## 16879 DESCRIPTION

16880 The *lpstat* utility writes to standard output information about the current status of the printer  
 16881 system.

16882 If no arguments are given, *lpstat* writes the status of all requests made to *lp* by the user that are  
 16883 still in the output queue.

## 16884 OPTIONS

16885 The *lpstat* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 16886 the option-arguments are optional and cannot be presented as separate arguments.

16887 Some of the options below can be followed by an optional *list* that can be in one of two forms: a  
 16888 list of items separated from one another by a comma, or a quoted list of items separated from  
 16889 one another by a comma or one or more blank characters, or combinations of both. See the  
 16890 **EXAMPLES** section.

16891 The omission of a *list* following such options causes all information relevant to the option to be  
 16892 written to standard output; for example:

16893 lpstat -u

16894 writes the status of all output requests that are still in the output queue.

16895 **-a[*list*]** Write the acceptance status of destinations for output requests. The *list* argument is a  
 16896 list of intermixed printer names and class names.

16897 **-c[*list*]** Write the class names and their members. The *list* argument is a list of class names.

16898 **-d** Write the system default destination for output requests.

16899 **-o[*list*]** Write the status of output requests. The *list* argument is a list of intermixed printer  
 16900 names, class names and *request IDs*.

16901 **-p[*list*]** Write the status of printers. The *list* argument is a list of printer names.

16902 **-r** Write the status of the printer request scheduler.

16903 **-s** Write a status summary, including the status of the printer scheduler, the system  
 16904 default destination, a list of class names and their members and a list of printers and  
 16905 their associated devices.

16906 **-t** Write all status information.

16907 **-u[*list*]** Write the status of output requests for users. The *list* argument is a list of login names.

16908 **-v[*list*]** Write the names of printers and the pathnames of the devices associated with them.  
 16909 The *list* argument is a list of printer names.

## 16910 OPERANDS

16911 The following operand is supported:

16912 *ID* A *request ID*, as returned by *lp*.

## 16913 STDIN

16914 Not used.



**16915 INPUT FILES**

16916       None.

**16917 ENVIRONMENT VARIABLES**

16918       The following environment variables affect the execution of *lpstat*:

16919       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 16920       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 16921       default locale will be used. If any of the internationalisation variables contains an  
 16922       invalid setting, the utility will behave as if none of the variables had been defined.

16923       **LC\_ALL**

16924       If set to a non-empty string value, override the values of all the other  
 16925       internationalisation variables.

16926       **LC\_CTYPE**

16927       Determine the locale for the interpretation of sequences of bytes of text data as  
 16928       characters (for example, single- as opposed to multi-byte characters in arguments).

16929       **LC\_MESSAGES**

16930       Determine the locale that should be used to affect the format and contents of diagnostic  
 16931       messages written to standard error, and informative messages written to standard  
 16932       output.

16933       **LC\_TIME**

16934       Determine the format of date and time strings output when displaying printer status  
 16935       information with the **-a**, **-o**, **-p**, **-t** or **-u** options.

16936 EX      **NLSPATH**

16937       Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

16938       **TZ**       Determine the timezone used with date and time strings.

**16939 ASYNCHRONOUS EVENTS**

16940       Default.

**16941 STDOUT**

16942       The standard output is a text file containing the information described in the **OPTIONS** section,  
 16943       in an unspecified format.

**16944 STDERR**

16945       Used only for diagnostic messages.

**16946 OUTPUT FILES**

16947       None.

**16948 EXTENDED DESCRIPTION**

16949       None.

**16950 EXIT STATUS**

16951       The following exit values are returned:

16952       0   Successful completion.

16953       >0 An error occurred.

**16954 CONSEQUENCES OF ERRORS**

16955       Default.

**16956 APPLICATION USAGE**

16957       The *lpstat* utility cannot reliably determine the status of print requests in all conceivable  
 16958       circumstances. When the printer is under the control of another operating system or resides on a

16959 remote system across a network, it need not be possible to determine the status of the print job  
 16960 after it has left the control of the local operating system. Even on local printers, spooling  
 16961 hardware in the printer may make it appear that the print job has been completed long before  
 16962 the final page is printed.

#### 16963 EXAMPLES

16964 1. Obtain the status of two printers, the pathnames of two printers, a list of all class names  
 16965 and the status of the request named **HiPri-33**:

16966 `lpstat -plaser1,laser4 -v"laser2 laser3" -c HiPri-33`

16967 OB 2. Obtain user print job status using the obsolescent mixed blank and comma form:

16968 `lpstat -u"ddg,gmv, maw"`

#### 16969 FUTURE DIRECTIONS

16970 A version of *lpstat* that fully supports the **XBD** specification, **Section 10.2, Utility Syntax**  
 16971 **Guidelines** may be introduced in a future issue.

#### 16972 SEE ALSO

16973 *cancel*, *lp*.

#### 16974 CHANGE HISTORY

16975 First released in Issue 2.

#### 16976 Issue 4

16977 Format reorganised.

16978 Exceptions to Utility Syntax Guidelines conformance noted.

16979 Internationalised environment variable support mandated.

#### 16980 Issue 5

16981 Marked LEGACY.

## 16982 NAME

16983       ls — list directory contents

## 16984 SYNOPSIS

16985 EX       ls [-CFRacdilqrutl][-fgmnopsx][file...]

## 16986 DESCRIPTION

16987       For each operand that names a file of a type other than directory, *ls* writes the name of the file as  
 16988       well as any requested, associated information. For each operand that names a file of type  
 16989       directory, *ls* writes the names of files contained within that directory, as well as any requested,  
 16990       associated information.

16991       If no operands are specified, the contents of the current directory are written. If more than one  
 16992       operand is specified, non-directory operands are written first; directory and non-directory  
 16993       operands are sorted separately according to the collating sequence in the current locale.

## 16994 OPTIONS

16995       The *ls* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

16996       The following options are supported:

16997       **-C**       Write multi-text-column output with entries sorted down the columns, according to the  
 16998       collating sequence. The number of text columns and the column separator characters  
 16999       are unspecified, but should be adapted to the nature of the output device.

17000       **-F**       Write a slash (/) immediately after each pathname that is a directory, an asterisk (\*)  
 17001 EX       after each that is executable, and a vertical bar (|) after each that is a FIFO. For other  
 17002       file types, other symbols may be written.

17003       **-R**       Recursively list subdirectories encountered.

17004       **-a**       Write out all directory entries, including those whose names begin with a period (.).  
 17005       Entries beginning with a period will not be written out unless explicitly referenced, the  
 17006       **-a** option is supplied, or an implementation-dependent condition causes them to be  
 17007       written.

17008       **-c**       Use time of last modification of the file status information (see <sys/stat.h> in the **XSH**  
 17009       specification) instead of last modification of the file itself for sorting (**-t**) or writing (**-l**).

17010       **-d**       Do not treat directories differently from other types of files. The use of **-d** with **-R**  
 17011       produces unspecified results.

17012 EX       **-f**       Force each argument to be interpreted as a directory and list the name found in each  
 17013       slot. This option turns off **-l**, **-t**, **-s** and **-r**, and turns on **-a**; the order is the order in  
 17014       which entries appear in the directory.

17015 EX       **-g**       The same as **-l**, except that the owner is not written.17016       **-i**       For each file, write the file's file serial number (see *stat()* in the **XSH** specification).

17017       **-l**       (The letter ell.) Write out in long format (see the **STDOUT** section). When **-l** (ell) is  
 17018       specified, **-1** (one) is assumed.

17019 EX       **-m**       Stream output format; list files across the page, separated by commas.

17020 EX       **-n**       The same as **-l**, except that the owner's UID and GID numbers are written, rather than  
 17021       the associated character strings.

|          |           |                                                                                                                                                                                                                                                                     |
|----------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17022 EX | <b>-o</b> | The same as <b>-l</b> , except that the group is not written.                                                                                                                                                                                                       |
| 17023 EX | <b>-p</b> | Write a slash (/) after each filename if that file is a directory.                                                                                                                                                                                                  |
| 17024    | <b>-q</b> | Force each instance of non-printable filename characters and tab characters to be written as the question-mark (?) character. Implementations may provide this option by default if the output is to a terminal device.                                             |
| 17025    |           |                                                                                                                                                                                                                                                                     |
| 17026    |           |                                                                                                                                                                                                                                                                     |
| 17027    | <b>-r</b> | Reverse the order of the sort to get reverse collating sequence or oldest first.                                                                                                                                                                                    |
| 17028 EX | <b>-s</b> | Indicate the total number of file system blocks consumed by each file displayed. The block size is implementation-dependent.                                                                                                                                        |
| 17029    |           |                                                                                                                                                                                                                                                                     |
| 17030    | <b>-t</b> | Sort by time modified (most recently modified first) before sorting the operands by the collating sequence.                                                                                                                                                         |
| 17031    |           |                                                                                                                                                                                                                                                                     |
| 17032    | <b>-u</b> | Use time of last access (see <sys/stat.h> in the XSH specification) instead of last modification of the file for sorting ( <b>-t</b> ) or writing ( <b>-l</b> ).                                                                                                    |
| 17033    |           |                                                                                                                                                                                                                                                                     |
| 17034 EX | <b>-x</b> | The same as <b>-C</b> , except that the multi-text-column output is produced with entries sorted across, rather than down, the columns.                                                                                                                             |
| 17035    |           |                                                                                                                                                                                                                                                                     |
| 17036    | <b>-1</b> | (The numeric digit one.) Force output to be one entry per line.                                                                                                                                                                                                     |
| 17037    |           | Specifying more than one of the options in the following mutually exclusive pairs is not considered an error: <b>-C</b> and <b>-l</b> (ell), <b>-m</b> and <b>-l</b> (ell), <b>-x</b> and <b>-l</b> (ell), <b>-C</b> and <b>-1</b> (one), <b>-c</b> and <b>-u</b> . |
| 17038 EX |           | The last option specified in each pair determines the output format.                                                                                                                                                                                                |
| 17039    |           |                                                                                                                                                                                                                                                                     |

#### 17040 OPERANDS

17041 The following operand is supported:

|       |             |                                                                                                                                |
|-------|-------------|--------------------------------------------------------------------------------------------------------------------------------|
| 17042 | <i>file</i> | A pathname of a file to be written. If the file specified is not found, a diagnostic message will be output on standard error. |
| 17043 |             |                                                                                                                                |

#### 17044 STDIN

17045 Not used.

#### 17046 INPUT FILES

17047 None.

#### 17048 ENVIRONMENT VARIABLES

17049 The following environment variables affect the execution of *ls*:

##### 17050 COLUMNS

17051 Determine the user's preferred column position width for writing multiple text-column output. If this variable contains a string representing a decimal integer, the *ls* utility calculates how many pathname text columns to write (see **-C**) based on the width provided. If *COLUMNS* is not set or invalid, an implementation-dependent number of column positions is assumed, based on the implementation's knowledge of the output device. The column width chosen to write the names of files in any given directory will be constant. Filenames will not be truncated to fit into the multiple text-column output.

17059 *LANG* Provide a default value for the internationalisation variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

##### 17063 LC\_ALL

17064 If set to a non-empty string value, override the values of all the other internationalisation variables.

17065

17066 **LC\_COLLATE**  
 17067 Determine the locale for character collation information in determining the pathname  
 17068 collation sequence.

17069 **LC\_CTYPE**  
 17070 Determine the locale for the interpretation of sequences of bytes of text data as  
 17071 characters (for example, single- versus multi-byte characters in arguments) and which  
 17072 characters are defined as printable (character class **print**).

17073 **LC\_MESSAGES**  
 17074 Determine the locale that should be used to affect the format and contents of diagnostic  
 17075 messages written to standard error.

17076 **LC\_TIME**  
 17077 Determine the format and contents for date and time strings written by *ls*.

17078 EX **NLSPATH**  
 17079 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

17080 **TZ** Determine the timezone for date and time strings written by *ls*.

17081 **ASYNCHRONOUS EVENTS**  
 17082 Default.

17083 **STDOUT**  
 17084 The default format is to list one entry per line to standard output; the exceptions are to terminals  
 17085 EX or when one of the **-C**, **-m** or **-x** options is specified. If the output is to a terminal, the format is  
 17086 implementation-dependent.

17087 EX When **-m** is specified, the format used is:  
 17088 `"%s, %s, ...\\n", <filename1>, <filename2>`  
 17089 where the largest number of filenames is written without exceeding the length of the line.

17090 If the **-i** option is specified, the file's file serial number (see <sys/stat.h> in the **XSH** specification)  
 17091 is written in the following format before any other output for the corresponding entry:  
 17092 `"%u ", <file serial number>`

17093 If the **-l** option is specified, the following information will be written:  
 17094 `"%s %u %s %s %u %s %s\\n", <file mode>, <number of links>,  
 17095 <owner name>, <group name>, <number of bytes in the file>,  
 17096 <date and time>, <pathname>`

17097 EX The **-g**, **-n** and **-o** options use the same format as **-l**, but with omitted items and their associated  
 17098 blank characters; see the **OPTIONS** section.

17099 EX If <owner name> or <group name> cannot be determined, or if **-n** is given, they are replaced with  
 17100 their associated numeric values using the format "%u".

17101 The <date and time>, field will contain the appropriate date and timestamp of when the file was  
 17102 last modified. In the POSIX locale, the field is the equivalent of the output of the following *date*  
 17103 command:  
 17104 `date "+%b %e %H:%M"`  
 17105 if the file has been modified in the last six months, or:  
 17106 `date "+%b %e %Y"`  
 17107 (where two space characters are used between %e and %Y) if the file has not been modified in

17108 the last six months or if the modification date is in the future, except that, in both cases, the final  
 17109 newline character produced by *date* is not included and the output is as if the *date* command  
 17110 were executed at the time of the last modification date of the file rather than the current time.  
 17111 When the LC\_TIME locale category is not set to the POSIX locale, a different format and order of  
 17112 presentation of this field may be used.

17113 If the file is a character special or block special file, the size of the file may be replaced with  
 17114 implementation-dependent information associated with the device in question.

17115 If the pathname was specified as a *file* operand, it will be written as specified.

17116 EX The file mode written under the **-l**, **-g**, **-n** and **-o** options consists of the following format:

17117     "%c%s%s%s%c", <entry type>, <owner permissions>,  
 17118     <group permissions>, <other permissions>,  
 17119     <optional alternate access method flag>

17120 The <optional alternate access method flag> is a single space character if there is no alternate or  
 17121 additional access control method associated with the file; otherwise, a printable character is  
 17122 used.

17123 The <entry type> character describes the type of file, as follows:

17124 d   Directory.  
 17125 b   Block special file.  
 17126 c   Character special file.  
 17127 p   FIFO.  
 17128 -   Regular file.

17129 Implementations may add other characters to this list to represent other, implementation-  
 17130 dependent, file types.

17131 The next three fields are three characters each:

17132 <owner permissions>  
 17133     Permissions for the file owner class (see **file access permissions** in the **XBD**  
 17134     specification, **Chapter 2, Glossary**).

17135 <group permissions>  
 17136     Permissions for the file group class.

17137 <other permissions>  
 17138     Permissions for the file other class.

17139 Each field has three character positions:

17140     1. If r, the file is readable; if "-", it is not readable.  
 17141     2. If w, the file is writable; if "-", it is not writable.  
 17142     3. The first of the following that applies:

17143         **S**   If in <owner permissions>, the file is not executable and set-user-ID mode is set. If in  
 17144         <group permissions>, the file is not executable and set-group-ID mode is set.

17145         **s**   If in <owner permissions>, the file is executable and set-user-ID mode is set. If in  
 17146         <group permissions>, the file is executable and set-group-ID mode is set.

17147         **x**   The file is executable or the directory is searchable.

17148         -   None of the attributes of S, s or x applies.

17149 Implementations may add other characters to this list for the third character position.  
 17150 Such additions will, however, be written in lower-case if the file is executable or  
 17151 searchable, and in upper-case if it is not.

17152 EX If any of the **-l**, **-g**, **-n**, **-o** or **-s** options is specified, each list of files within the directory will be  
 17153 preceded by a status line indicating the number of file system blocks occupied by files in the  
 17154 directory in 512-byte units, rounded up to the next integral number of units, if necessary. In the  
 17155 POSIX locale, the format is:

17156 "total %u\n", *<number of units in the directory>*

17157 If more than one directory, or a combination of non-directory files and directories are written,  
 17158 either as a result of specifying multiple operands, or the **-R** option, each list of files within a  
 17159 directory will be preceded by:

17160 "\n%s:\n", *<directory name>*

17161 If this string is the first thing to be written, the first newline character is not written. This output  
 17162 precedes the number of units in the directory.

17163 EX If the **-s** option is given, each file shall be written with the number of blocks used by the file.  
 17164 Along with **-C**, **-l**, **-m** or **-x**, the number and a space character precede the filename; with **-g**, **-l**,  
 17165 **-n** or **-o**, they precede each line describing a file.

#### 17166 **STDERR**

17167 Used only for diagnostic messages.

#### 17168 **OUTPUT FILES**

17169 None.

#### 17170 **EXTENDED DESCRIPTION**

17171 None.

#### 17172 **EXIT STATUS**

17173 The following exit values are returned:

17174 0 All files were written successfully.

17175 >0 An error occurred.

#### 17176 **CONSEQUENCES OF ERRORS**

17177 Default.

#### 17178 **APPLICATION USAGE**

17179 Many implementations use the equal sign (=) and the at sign (@) to denote sockets bound to the  
 17180 file system and symbolic links, respectively, for the **-F** option. Similarly, many historical  
 17181 implementations use the s character and the l character to denote sockets and symbolic links,  
 17182 respectively, as the entry type characters for the **-l** option.

17183 It is difficult for an application to use every part of the file modes field of **ls -l** in a portable  
 17184 manner. Certain file types and executable bits are not guaranteed to be exactly as shown, as  
 17185 implementations may have extensions. Applications can use this field to pass directly to a user  
 17186 printout or prompt, but actions based on its contents should generally be deferred, instead, to  
 17187 the *test* utility.

17188 The output of **ls** (with the **-l** and related options) contains information that logically could be  
 17189 used by utilities such as *chmod* and *touch* to restore files to a known state. However, this  
 17190 information is presented in a format that cannot be used directly by those utilities or be easily  
 17191 translated into a format that can be used. A character has been added to the end of the  
 17192 permissions string so that applications will at least have an indication that they may be working  
 17193 in an area they do not understand instead of assuming that they can translate the permissions

17194 string into something that can be used. Future issues or related documents may define one or  
 17195 more specific characters to be used based on different standard additional or alternative access  
 17196 control mechanisms.

17197 As with many of the utilities that deal with filenames, the output of *ls* for multiple files or in one  
 17198 of the long listing formats must be used carefully on systems where filenames can contain  
 17199 embedded white space. Systems and system administrators should institute policies and user  
 17200 training to limit the use of such filenames.

17201 The number of disk blocks occupied by the file that it reports varies depending on underlying  
 17202 file system type, block size units reported and the method of calculating the number of blocks.  
 17203 On some file system types, the number is the actual number of blocks occupied by the file  
 17204 (counting indirect blocks and ignoring holes in the file); on others it is calculated based on the  
 17205 file size (usually making an allowance for indirect blocks, but ignoring holes).

#### 17206 EXAMPLES

17207 An example of a small directory tree being fully listed with *ls -laRF a* in the POSIX locale:

```
17208 total 11
17209 drwxr-xr-x 3 hlj prog 64 Jul 4 12:07 ./
17210 drwxrwxrwx 4 hlj prog 3264 Jul 4 12:09 ../
17211 drwxr-xr-x 2 hlj prog 48 Jul 4 12:07 b/
17212 -rwxr--r-- 1 hlj prog 572 Jul 4 12:07 foo*

17213 a/b:
17214 total 4
17215 drwxr-xr-x 2 hlj prog 48 Jul 4 12:07 ./
17216 drwxr-xr-x 3 hlj prog 64 Jul 4 12:07 ../
17217 -rw-r--r-- 1 hlj prog 700 Jul 4 12:07 bar
```

#### 17218 FUTURE DIRECTIONS

17219 The *-s* uses implementation-dependent units and cannot be used portably; it may be withdrawn  
 17220 in a future issue.

17221 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 17222 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 17223 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 17224 finalised.

#### 17225 SEE ALSO

17226 *chmod*, *find*, the XSH specification description of <sys/stat.h>.

#### 17227 CHANGE HISTORY

17228 First released in Issue 2.

#### 17229 Issue 4

17230 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 17231 Issue 5

17232 Second FUTURE DIRECTION added.



17233 **NAME**17234       m4 — macro processor (**DEVELOPMENT**)17235 **SYNOPSIS**17236 EX       m4 [-s] [ -D *name*[=*val*]]... [-U *name*]... *file*...17237 **DESCRIPTION**

17238       The *m4* utility is a macro processor that reads one or more text files, processes them according to  
 17239       their included macro statements, and writes the results to standard output.

17240 **OPTIONS**

17241       The *m4* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 17242       that the order of the **-D** and **-U** options is significant. The following options are supported:

17243       **-s**       Enable line synchronisation output for the *c89* preprocessor phase (that is, **#line**  
 17244       directives).

17245       **-D *name*[=*val*]**  
 17246       Define *name* to *val* or to null if *=val* is omitted.

17247       **-U *name***  
 17248       Undefine *name*.

17249 **OPERANDS**

17250       The following operand is supported:

17251       *file*       A pathname of a text file to be processed. If no *file* is given, or if it is "-", the standard  
 17252       input is read.

17253 **STDIN**17254       The standard input is a text file that is used if no *file* operand is given, or if it is "-".17255 **INPUT FILES**17256       The input file named by the *file* operand is a text file.17257 **ENVIRONMENT VARIABLES**17258       The following environment variables affect the execution of *m4*:

17259       **LANG**     Provide a default value for the internationalisation variables that are unset or null. If  
 17260       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 17261       default locale will be used. If any of the internationalisation variables contains an  
 17262       invalid setting, the utility will behave as if none of the variables had been defined.

17263       **LC\_ALL**  
 17264       If set to a non-empty string value, override the values of all the other  
 17265       internationalisation variables.

17266       **LC\_CTYPE**  
 17267       Determine the locale for the interpretation of sequences of bytes of text data as  
 17268       characters (for example, single- as opposed to multi-byte characters in arguments and  
 17269       input files).

17270       **LC\_MESSAGES**  
 17271       Determine the locale that should be used to affect the format and contents of diagnostic  
 17272       messages written to standard error.

17273       **NLSPATH**  
 17274       Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

17275 **ASYNCHRONOUS EVENTS**

17276 Default.

17277 **STDOUT**

17278 The standard output is the same as the input files, after being processed for macro expansion.

17279 **STDERR**

17280 Used to display strings with the **errprint** macro, macro tracing enabled by the **traceon** macro, the  
 17281 defined text for macros written by the **dumpdef** macro, or for diagnostic messages.

17282 **OUTPUT FILES**

17283 None.

17284 **EXTENDED DESCRIPTION**

17285 The *m4* utility compares each token from the input against the set of built-in and user-defined  
 17286 macros. If the token matches the name of a macro, then the token is replaced by the macros  
 17287 defining text, if any, and rescanned for matching macro names. Once no portion of the token  
 17288 matches the name of a macro, it is written to standard output. Macros may have arguments, in  
 17289 which case the arguments will be substituted into the defining text before it is rescanned.

17290 Macro calls have the form:

17291 `name(arg1, arg2, . . . , argn)`

17292 Macro names consist of letters, digits and underscores, where the first character is not a digit.  
 17293 Tokens not of this form are not treated as macro names.

17294 The left parenthesis must immediately follow the name of the macro. If a token matching the  
 17295 name of a macro is not followed by a left parenthesis, it will be handled as a use of that macro  
 17296 without arguments.

17297 If a macro name is followed by a left parenthesis, its arguments are the comma-separated tokens  
 17298 between the left parenthesis and the matching right parenthesis. Unquoted blank and newline  
 17299 characters preceding each argument are ignored. All other characters, including trailing blank  
 17300 and newline characters, are retained. Commas enclosed between left and right parenthesis  
 17301 characters do not delimit arguments.

17302 Arguments are positionally defined and referenced. The string \$1 in the defining text will be  
 17303 replaced by the first argument. Systems support at least nine arguments; only the first nine can  
 17304 be referenced, using the strings \$1 to \$9, inclusive. The string \$0 will be replaced with the name  
 17305 of the macro. The string \$# will be replaced by the number of arguments as a string. The string  
 17306 \$\* will be replaced by a list of all of the arguments, separated by commas. The string @\$ will be  
 17307 replaced by a list of all of the arguments separated by commas, and each argument will be  
 17308 quoted using the current left and right quoting strings.

17309 If fewer arguments are supplied than are in the macro definition, the omitted arguments are  
 17310 taken to be null. It is not an error if more arguments are supplied than are in the macro  
 17311 definition.

17312 No special meaning is given to any characters enclosed between matching left and right quoting  
 17313 strings, but the quoting strings are themselves discarded. By default, the left quoting string  
 17314 consists of a grave accent (`) and the right quoting string consists of an acute accent (') see also  
 17315 the **changequote** macro.

17316 Comments are written but not scanned for matching macro names; by default, the begin-  
 17317 comment string consists of the number sign character and the end-comment string consists of a  
 17318 newline character. See also the **changecom** and **dnl** macros.

17319 The *m4* utility makes available the following built-in macros. They can be redefined, but once  
 17320 this is done the original meaning is lost. Their values are null unless otherwise stated.

17321 **changeocom**  
 17322 The **changeocom** macro sets the begin- and end-comment strings. With no arguments,  
 17323 the comment mechanism is disabled. With a single argument, that argument becomes  
 17324 the begin-comment string and the newline character becomes the end-comment string.  
 17325 With two arguments, the first argument becomes the begin-comment string and the  
 17326 second argument becomes the end-comment string. Systems support comment strings  
 17327 of at least five characters.

17328 **changequote**  
 17329 The **changequote** macro sets the begin- and end-quote strings. With no arguments, the  
 17330 quote strings are set to the default values (that is, `'). With a single argument, that  
 17331 argument becomes the begin-quote string and the newline character becomes the end-  
 17332 quote string. With two arguments, the first argument becomes the begin-quote string  
 17333 and the second argument becomes the end-quote string. Systems support quote strings  
 17334 of at least five characters.

17335 **decr** The defining text of the **decr** macro is its first argument decremented by 1. It is an error  
 17336 to specify an argument containing any non-numeric characters.

17337 **define** The second argument is specified as the defining text of the macro whose name is the  
 17338 first argument.

17339 **defn** The defining text of the **defn** macro is the quoted definition (using the current quoting  
 17340 strings) of its arguments.

17341 **divert** The *m4* utility maintains ten temporary buffers, numbered 0 to 9, inclusive. When the  
 17342 last of the input has been processed, any output that has been placed in these buffers  
 17343 will be written to standard output in buffer-numerical order. The **divert** macro diverts  
 17344 future output to the buffer specified by its argument. Specifying no argument or an  
 17345 argument of 0 resumes the normal output process. Output diverted to a stream other  
 17346 than 0 to 9 is discarded. It is an error to specify an argument containing any non-  
 17347 numeric characters.

17348 **divnum** The defining text of the **divnum** macro is the number of the current output stream as a  
 17349 string.

17350 **dnl** The **dnl** macro causes *m4* to discard all input characters up to and including the next  
 17351 newline character.

17352 **dumpdef**  
 17353 The **dumpdef** macro writes the defined text to standard error for each of the macros  
 17354 specified as arguments, or, if no arguments are specified, for all macros.

17355 **errprint** The **errprint** macro writes its arguments to standard error.

17356 **eval** The **eval** macro evaluates its first argument as an arithmetic expression, using 32-bit  
 17357 signed integer arithmetic. All of the C-language operators are supported, except for [],  
 17358 ->, ++, --, (type), unary \*, sizeof, ":", "?:", and all assignment operators. It is an error  
 17359 to specify any of these operators. Precedence and associativity are as in C. Systems  
 17360 support octal and hexadecimal numbers as in C. The second argument, if specified,  
 17361 sets the radix for the result; the default is 10. The third argument, if specified, sets the  
 17362 minimum number of digits in the result. It is an error to specify an argument  
 17363 containing any non-numeric characters.

|       |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17364 | <b>ifdef</b>    | If the first argument to the <b>ifdef</b> macro is defined, the defining text is the second argument. Otherwise, the defining text is the third argument, if specified, or the null string, if not.                                                                                                                                                                                                                                                                                                                                                                      |
| 17365 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17366 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17367 | <b>ifelse</b>   | If the first argument (or the defining text of the first argument if it is a macro name) to the <b>ifelse</b> macro is the same as the second argument (or the defining text of the second argument if it is a macro name), then the defining text is the third argument. If there are more than four arguments, the initial comparison of the first and second arguments are repeated for each group of three arguments. If no match is found, the defining text will be the argument following the last set of three compared, otherwise it will be null.              |
| 17368 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17369 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17370 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17371 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17372 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17373 | <b>include</b>  | The defining text for the <b>include</b> macro is the contents of the file named by the first argument. It is an error if the file cannot be read.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 17374 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17375 | <b>incr</b>     | The defining text of the <b>incr</b> macro is its first argument incremented by 1. It is an error to specify an argument containing any non-numeric characters.                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17376 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17377 | <b>index</b>    | The defining text of the <b>index</b> macro is the first character position (as a string) in the first argument where a string matching the second argument begins (zero origin), or -1 if the second argument does not occur.                                                                                                                                                                                                                                                                                                                                           |
| 17378 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17379 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17380 | <b>len</b>      | The defining text of the <b>len</b> macro is the length (as a string) of the first argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 17381 | <b>m4exit</b>   | Exit from the <i>m4</i> utility. If the first argument is specified, it will be the exit code. The default is zero. It is an error to specify an argument containing any non-numeric characters.                                                                                                                                                                                                                                                                                                                                                                         |
| 17382 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17383 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17384 | <b>m4wrap</b>   | The first argument will be processed when EOF is reached. If the <b>m4wrap</b> macro is used multiple times, the arguments specified will be processed in the order in which the <b>m4wrap</b> macros were processed.                                                                                                                                                                                                                                                                                                                                                    |
| 17385 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17386 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17387 | <b>maketemp</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17388 |                 | The defining text is the first argument, with any trailing capital X characters replaced with the current process ID as a string.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 17389 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17390 | <b>popdef</b>   | The <b>popdef</b> macro deletes the current definition of its arguments, replacing it with the previous one. If there is no previous definition, the macro is undefined.                                                                                                                                                                                                                                                                                                                                                                                                 |
| 17391 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17392 | <b>pushdef</b>  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17393 |                 | The <b>pushdef</b> macro is identical to the <b>define</b> macro with the exception that it preserves any current definition for future retrieval using the <b>popdef</b> macro.                                                                                                                                                                                                                                                                                                                                                                                         |
| 17394 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17395 | <b>shift</b>    | The defining text for the <b>shift</b> macro is all of its arguments except for the first one.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 17396 | <b>sinclude</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17397 |                 | The <b>sinclude</b> macro is identical to the <b>include</b> macro, except that it is not an error if the file is inaccessible.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17398 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17399 | <b>substr</b>   | The defining text for the <b>substr</b> macro is the substring of the first argument beginning at the zero-offset character position specified by the second argument. The third argument, if specified, is the number of characters to select; if not specified, the characters from the starting point to the end of the first argument become the defining text. It is not an error to specify a starting point beyond the end of the first argument and the defining text will be null. It is an error to specify an argument containing any non-numeric characters. |
| 17400 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17401 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17402 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17403 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17404 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17405 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17406 | <b>syscmd</b>   | The <b>syscmd</b> macro interprets its first argument as a shell command line. The defining text is the string result of that command. No output redirection is performed by the <i>m4</i> utility. The exit status value from the command can be retrieved using the <b>sysval</b>                                                                                                                                                                                                                                                                                      |
| 17407 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17408 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

17409 macro.

17410 **sysval** The defining text of the **sysval** macro is the exit value of the utility last invoked by the  
 17411 **syscmd** macro (as a string).

17412 **traceon** The **traceon** macro enables tracing for the macros specified as arguments, or, if no  
 17413 arguments are specified, for all macros. The trace output is written to standard error in  
 17414 an unspecified format.

17415 **traceoff** The **traceoff** macro disables tracing for the macros specified as arguments, or, if no  
 17416 arguments are specified, for all macros.

17417 **translit** The defining text of the **translit** macro is the first argument with every character that  
 17418 occurs in the second argument replaced with the corresponding character from the  
 17419 third argument.

17420 **undefine**  
 17421 The **undefine** macro deletes all definitions (including those preserved using the  
 17422 **pushdef** macro) of the macros named by its arguments.

17423 **undivert**  
 17424 The **undivert** macro causes immediate output of any text in temporary buffers named  
 17425 as arguments, or all temporary buffers if no arguments are specified. Buffers can be  
 17426 undiverted into other temporary buffers. Undiverting discards the contents of the  
 17427 temporary buffer. It is an error to specify an argument containing any non-numeric  
 17428 characters.

17429 **EXIT STATUS**  
 17430 The following exit values are returned:

17431 0 Successful completion.  
 17432 >0 An error occurred

17433 If the **m4exit** macro is used, the exit value can be specified by the input file.

17434 **CONSEQUENCES OF ERRORS**  
 17435 Default.

17436 **APPLICATION USAGE**  
 17437 The **defn** macro is useful for renaming macros, especially built-ins.

17438 **EXAMPLES**  
 17439 An example of a single *m4* input file capable of generating two output files follows. The file  
 17440 **file1.m4** could contain lines such as:

17441 `if(VER, 1, do_something)`  
 17442 `if(VER, 2, do_something)`

17443 The makefile for the program might include:

17444 `file1.1.c : file1.m4`  
 17445 `m4 -D VER=1 file1.m4 > file1.1.c`  
 17446 `...`  
 17447 `file1.2.c : file1.m4`  
 17448 `m4 -D VER=2 file1.m4 > file1.2.c`  
 17449 `...`

17450 The **-U** option can be used to undefine **VER**. If **file1.m4** contains:

```
17451 if(VER, 1, do_something)
17452 if(VER, 2, do_something)
17453 ifndef(VER, do_something)
```

17454 then the makefile would contain:

```
17455 file1.0.c : file1.m4
17456 m4 -U VER file1.m4 > file1.0.c
17457 ...
17458 file1.1.c : file1.m4
17459 m4 -D VER=1 file1.m4 > file1.1.c
17460 ...
17461 file1.2.c : file1.m4
17462 m4 -D VER=2 file1.m4 > file1.2.c
17463 ...
```

#### 17464 **FUTURE DIRECTIONS**

17465 None.

#### 17466 **SEE ALSO**

17467 *c89*.

#### 17468 **CHANGE HISTORY**

17469 First released in Issue 2.

#### 17470 **Issue 4**

17471 Format reorganised.

17472 Utility Syntax Guideline support mandated.

17473 Internationalised environment variable support mandated.

#### 17474 **Issue 5**

17475 Added the phrase “the defined text for macros written by the **dumpdef** macro”, to the  
 17476 description of **STDERR**, and the description of **dumpdef** is updated to indicate that output is  
 17477 written to standard error. The description of **eval** is updated to indicate that the list of excluded  
 17478 C operators includes unary “&” and “.”. In the description of **ifdef**, the phrase “and it is not  
 17479 defined to be zero” is deleted.

17480 **NAME**17481       mail — send or read mail (**LEGACY**)17482 **SYNOPSIS**17483 EX       mail [-e] [-f *file*]17484 UN EX     mail [-e | -p] [-qr] [-f *file*]17485 UN EX     mail [-t] *name* ...17486 **DESCRIPTION**

17487       The *mail* utility cannot guarantee support for all character encodings in all circumstances. For  
 17488       example, inter-system mail may be restricted to 7-bit data by the underlying network, 8-bit data  
 17489       need not be portable to non-internationalised systems, and so on. Under these circumstances, it  
 17490       is recommended that only characters defined in the ISO/IEC 646:1991 standard International  
 17491       Reference Version (equivalent to ASCII) 7-bit range of characters be used.

17492       **Reading Mail**

17493       The *mail* utility without arguments writes a user's mail to standard output, message-by-  
 17494       message. Mail is stored in the user's individual mailfile. For each message, the user is given a  
 17495       prompt and a line is read from the standard input to determine the disposition of the message;  
 17496       see the EXTENDED DESCRIPTION section.

17497       **Sending Mail**

17498       When *names* (user login names) are given, *mail* takes the standard input up to an end-of-file (or  
 17499       up to a line consisting of just a '.') and adds it to each user's mailfile. The message is preceded  
 17500       by the sender's name and a postmark. Lines in the message that begin with the sequence **From**  
 17501       are preceded with a ">".

17502       If a user being sent mail is not recognised, or if *mail* is interrupted during input, the message is  
 17503       saved in the file **dead.letter** to allow editing and resending. Note that this is regarded as a  
 17504       temporary file in that it is recreated every time it is needed, erasing the previous contents of  
 17505       **dead.letter**. The *mail* utility tries to create **dead.letter** in the current directory. If that fails, it tries  
 17506       to create **dead.letter** in the directory specified by the *HOME* environment variable.

17507       It may also be possible to send mail to remote systems using system-specific naming  
 17508       conventions.

17509       There are implementation-dependent mechanisms that can be used to cause all mail sent to the  
 17510       user to be forwarded to one or more other destinations.

17511 **OPTIONS**

17512       The *mail* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 17513       following options are supported for reading mail:

17514       **-e**       Test for the presence of mail. Display nothing and exit with a successful return code if  
 17515       there is mail to read.

17516 UN       **-p**       Write all mail to standard output without prompting for disposition.

17517 UN       **-q**       Terminate after interrupts. By default, an interrupt terminates only the message being  
 17518       written.

17519 UN       **-r**       Write messages in first-in, first-out order.

17520        **-f file**    Use *file* (for example, **mbox**) instead of the default mailfile.

17521        The following option is supported for sending mail:

17522 UN     **-t**        Precede the message by a list of all users the *mail* is sent to.

### 17523 OPERANDS

17524        The following operand is supported for sending mail:

17525        *name*     A user login name.

### 17526 STDIN

17527        The standard input is a text file of commands for writing mail or a text file to be added to the  
17528        user's mailfile when sending mail.

### 17529 INPUT FILES

17530        None.

### 17531 ENVIRONMENT VARIABLES

17532        The following environment variable affects the execution of *mail*:

17533        *HOME*    Determine the pathname of the user's home directory.

17534        The following environment variables may affect the execution of *mail*:

17535        *LANG*    Provide a default value for the internationalisation variables that are unset or null. If  
17536        *LANG* is unset or null, the corresponding value from the implementation-dependent  
17537        default locale will be used. If any of the internationalisation variables contains an  
17538        invalid setting, the utility will behave as if none of the variables had been defined.

17539        *LC\_ALL*

17540        If set to a non-empty string value, override the values of all the other  
17541        internationalisation variables.

17542        *LC\_CTYPE*

17543        Determine the locale for the interpretation of sequences of bytes of text data as  
17544        characters (for example, single- as opposed to multi-byte characters in arguments and  
17545        input files).

17546        *LC\_MESSAGES*

17547        Determine the locale that should be used to affect the format and contents of diagnostic  
17548        messages written to standard error, and informative messages written to standard  
17549        output.

17550        *LC\_TIME*

17551        Determine the format and contents of date and time strings displayed by the *mail*  
17552        utility.

17553        *NLSPATH*

17554        Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

17555        *TZ*        Determine the timezone used with date and time strings.

### 17556 ASYNCHRONOUS EVENTS

17557        When reading mail, signals are caught and the user is returned immediately to the next prompt;  
17558        however, if the **-q** option is specified, *mail* terminates. When sending mail, signals are caught,  
17559        interrupting the user's input. Any text already input is saved in the file **dead.letter**.

### 17560 STDOUT

17561        When reading mail, the standard output is used for prompting and writing mail messages; the  
17562        format of prompting and other informational messages is unspecified. When sending mail, the



17563 standard output is not used.

#### 17564 **STDERR**

17565 Used only for diagnostic messages.

#### 17566 **OUTPUT FILES**

17567 When reading mail, messages can be appended to files designated by the **s** or **w** commands or to  
 17568 a user's mailfile by using the **m** command. When sending mail, messages are written to mailfiles  
 17569 or to **dead.letter**.

#### 17570 **EXTENDED DESCRIPTION**

17571 When reading mail, the following commands read from the standard input determine the  
 17572 disposition of messages:

17573 newline

17574 Go on to next message.

17575 + Same as the newline character.

17576 **d** Delete message and go on to next message.

17577 **p** Display message again.

17578 – Go back to previous message.

17579 **s[ file ]** Save the message in the named *file* (**mbox** is default).

17580 **w[ file ]** Save the message (on some implementations, without its header) in the named *file*  
 17581 (**mbox** is default).

17582 **m[ name ... ]**

17583 Mail the message to the named users; the default is the user who invoked *mail*.

17584 **q** Store undeleted mail and stop.

17585 <EOF> Same as **q**.

17586 **x** Put all mail back in the mailfile unchanged and stop.

17587 **!command**

17588 Escape to the command interpreter to execute *command*.

17589 **\*** Display a command summary.

#### 17590 **EXIT STATUS**

17591 The following exit values are returned:

17592 0 Successful completion when the user had mail.

17593 1 The user had no mail or an initialisation error occurred.

17594 >1 An error occurred after initialisation.

#### 17595 **CONSEQUENCES OF ERRORS**

17596 When reading mail, the mailfile is unchanged. When sending mail, some of the named users  
 17597 need not have their mailfiles appended with the message.

#### 17598 **APPLICATION USAGE**

17599 Delivery of messages to remote systems requires the existence of communication paths to such  
 17600 systems. These need not exist.

17601 The location of stored mail on exiting from *mail* using the **q** command differs between  
 17602 implementations and may be either the user's mailfile or the user's **mbox**.

|       |                                                                                                 |  |
|-------|-------------------------------------------------------------------------------------------------|--|
| 17603 | In the description of reading mail, the phrase “go on to next message” might or might not imply |  |
| 17604 | the displaying of the next message.                                                             |  |
| 17605 | Input lines are limited to {LINE_MAX} bytes, but mailers between systems may impose more        |  |
| 17606 | severe line-length restrictions.                                                                |  |
| 17607 | Applications should migrate to the <i>mailx</i> utility.                                        |  |
| 17608 | <b>EXAMPLES</b>                                                                                 |  |
| 17609 | None.                                                                                           |  |
| 17610 | <b>FUTURE DIRECTIONS</b>                                                                        |  |
| 17611 | None.                                                                                           |  |
| 17612 | <b>SEE ALSO</b>                                                                                 |  |
| 17613 | <i>mailx</i> , <i>uuencode</i> .                                                                |  |
| 17614 | <b>CHANGE HISTORY</b>                                                                           |  |
| 17615 | First released in Issue 2.                                                                      |  |
| 17616 | <b>Issue 4</b>                                                                                  |  |
| 17617 | Format reorganised.                                                                             |  |
| 17618 | Marked TO BE WITHDRAWN.                                                                         |  |
| 17619 | <b>Issue 5</b>                                                                                  |  |
| 17620 | Marked LEGACY.                                                                                  |  |

17621 **NAME**

17622           mailx — process messages

17623 **SYNOPSIS**17624           **Send Mode:**17625           mailx [-s *subject*] *address...*17626           **Receive Mode:**

17627           mailx -e

17628 EX       mailx [-HiNn] [-F] [-u *user*]17629 EX       mailx -f [-HiNn] [-F] [*file*]17630 **DESCRIPTION**

17631       The *mailx* utility provides a message sending and receiving facility. It has two major modes,  
 17632       selected by the options used: Send Mode and Receive Mode.

17633           **Send Mode**

17634       Send Mode can be used by applications or users to send messages from the text in standard  
 17635       input.

17636           **Receive Mode**

17637       Receive Mode is more oriented to interactive users. Mail can be read and sent in this interactive  
 17638       mode.

17639       When reading mail, *mailx* provides commands to facilitate saving, deleting and responding to  
 17640       messages. When sending mail, *mailx* allows editing, reviewing and other modification of the  
 17641       message as it is entered.

17642       Incoming mail is stored in one or more unspecified locations for each user, collectively called the  
 17643       system *mailbox* for that user. When *mailx* is invoked in Receive Mode, the system mailbox is the  
 17644       default place to find them. As messages are read, they will be marked to be moved to a  
 17645       secondary file for storage, unless specific action is taken. This secondary file is called the **mbox**  
 17646       and is normally located in the *HOME* directory of the user (see *MBOX* in the ENVIRONMENT  
 17647       VARIABLES section for a description of this file). Messages remain in this file until explicitly  
 17648       removed. When the *-f* option is used to read mail messages from secondary files, messages will  
 17649       be retained in those files unless specifically removed. All three of these locations system  
 17650       mailbox, **mbox** and secondary file are referred to in this section as simply “mailboxes”, unless  
 17651       more specific identification is required.

17652 **OPTIONS**

17653       The *mailx* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

17654       The following options are supported:

17655       **-e**       Test for the presence of mail in the system mailbox. The *mailx* utility will write nothing  
 17656       and exit with a successful return code if there is mail to read.

17657       **-f**       Read messages from the file named by the *file* operand instead of the system mailbox.  
 17658       (See also **folder**.) If no *file* operand is specified, read messages from the **mbox** instead  
 17659       of the system mailbox.

17660 EX     **-F**     Record the message in a file named after the first recipient. The name is the login-name portion of the address found first on the To: line in the mail header. Overrides the **record** variable, if set (see **Internal Variables in mailx** on page 489.)

17661

17662

17663     **-H**     Write a header summary only.

17664     **-i**     Ignore interrupts. (See also **ignore**).

17665     **-n**     Do not initialise from the system default start-up file. See the EXTENDED DESCRIPTION section.

17666

17667     **-N**     Do not write an initial header summary.

17668     **-s subject**  
               Set the Subject header field to *subject*. All characters in the *subject* string will appear in the delivered message. The results are unspecified if *subject* is longer than {LINE\_MAX} – 10 bytes or contains a newline character.

17669

17670

17671

17672     **-u user**  
               Read the system mailbox of the login name *user*. This will only be successful if the invoking user has the appropriate privileges to read the system mailbox of that user.

17673

17674

17675 **OPERANDS**

17676     The following operands are supported:

17677     *address*   Addressee of message. When **-n** is specified and no user start-up files are accessed (see the EXTENDED DESCRIPTION section), this must be an address to pass to the mail delivery system. Any system or user start-up files may enable aliases (see **alias** under **Commands in mailx** on page 492) that may modify the form of *address* before it is passed to the mail delivery system.

17678

17679

17680

17681

17682     *file*     A pathname of a file to be read instead of the system mailbox when **-f** is specified. The meaning of the *file* option-argument is affected by the contents of the **folder** internal variable; see **Internal Variables in mailx** on page 489.

17683

17684

17685 **STDIN**

17686     When *mailx* is invoked in Send Mode (the first synopsis line), standard input must be the message to be delivered to the specified addresses. In both Send and Receive Modes, standard-input lines beginning with the escape character (usually tilde (~)) affect processing as described in **Command Escapes in mailx** on page 499.

17687

17688

17689

17690 **INPUT FILES**

17691     When *mailx* is used as described by this specification, the *file* option-argument (see the **-f** option) and the **mbox** must be text files containing mail messages, formatted as described in the OUTPUT FILES section. The nature of the system mailbox is unspecified; it need not be a file.

17692

17693

17694 **ENVIRONMENT VARIABLES**

17695     The following environment variables affect the execution of *mailx*:

17696     **DEAD**    Determine the pathname of the file in which to save partial messages in case of interrupts or delivery errors. The default is **dead.letter** in the directory named by the *HOME* variable.

17697

17698

17699     **EDITOR**  
               Determine the name of a utility to invoke when the **edit** (see **Commands in mailx** on page 492) or **~e** (see **Command Escapes in mailx** on page 499) command is used. The default editor is *ed*.

17700

17701

17702 EX

17703     **HOME**    Determine the pathname of the user's home directory.

|          |                    |                                                                                                       |
|----------|--------------------|-------------------------------------------------------------------------------------------------------|
| 17704    | <b>LANG</b>        | Provide a default value for the internationalisation variables that are unset or null. If             |
| 17705    |                    | <b>LANG</b> is unset or null, the corresponding value from the implementation-dependent               |
| 17706    |                    | default locale will be used. If any of the internationalisation variables contains an                 |
| 17707    |                    | invalid setting, the utility will behave as if none of the variables had been defined.                |
| 17708    | <b>LC_ALL</b>      |                                                                                                       |
| 17709    |                    | If set to a non-empty string value, override the values of all the other                              |
| 17710    |                    | internationalisation variables.                                                                       |
| 17711    | <b>LC_CTYPE</b>    |                                                                                                       |
| 17712    |                    | Determine the locale for the interpretation of sequences of bytes of text data as                     |
| 17713    |                    | characters (for example, single- as opposed to multi-byte characters in arguments and                 |
| 17714    |                    | input files) and the handling of case-insensitive address and header-field comparisons.               |
| 17715    | <b>LC_TIME</b>     |                                                                                                       |
| 17716    |                    | Determine the format and contents of the date and time strings written by <i>mailx</i> .              |
| 17717    | <b>LC_MESSAGES</b> |                                                                                                       |
| 17718    |                    | Determine the locale that should be used to affect the format and contents of diagnostic              |
| 17719    |                    | messages written to standard error and informative messages written to standard                       |
| 17720    |                    | output.                                                                                               |
| 17721    | <b>LISTER</b>      | Determine a string representing the command for writing the contents of the <b>folder</b>             |
| 17722    |                    | directory to standard output when the <b>folders</b> command is given (see <b>folders</b> in          |
| 17723    |                    | <b>Commands in mailx</b> on page 492). Any string acceptable as a <i>command_string</i> operand       |
| 17724    |                    | to the <i>sh -c</i> command is valid. If this variable is null or not set, the output command         |
| 17725    |                    | will be <i>ls</i> . The default value is unset.                                                       |
| 17726    | <b>MAILRC</b>      |                                                                                                       |
| 17727    |                    | Determine the pathname of the start-up file. The default is <b>.mailrc</b> in the <i>HOME</i>         |
| 17728    |                    | directory.                                                                                            |
| 17729    | <b>MBOX</b>        | Determine a pathname of the file to save messages from the system mailbox that have                   |
| 17730    |                    | been read. The <b>exit</b> command overrides this function, as will saving the message                |
| 17731    |                    | explicitly in another file. The default is <b>mbox</b> in the directory named by the <i>HOME</i>      |
| 17732    |                    | variable.                                                                                             |
| 17733 EX | <b>NLSPATH</b>     |                                                                                                       |
| 17734    |                    | Determine the location of message catalogues for the processing of <b>LC_MESSAGES</b> .               |
| 17735    | <b>PAGER</b>       | Determine a string representing an output filtering or pagination command for writing                 |
| 17736    |                    | the output to the terminal. Any string acceptable as a <i>command_string</i> operand to the <i>sh</i> |
| 17737    |                    | <i>-c</i> command is valid. When standard output is a terminal device, the message output             |
| 17738    |                    | will be piped through the command if the <i>mailx</i> internal variable <b>crt</b> is set to a value  |
| 17739    |                    | less the number of lines in the message; see <b>Internal Variables in mailx</b> on page 489. If       |
| 17740    |                    | the <b>PAGER</b> variable is null or not set, the paginator will be either <i>more</i> or another     |
| 17741    |                    | paginator utility documented in the system documentation.                                             |
| 17742    | <b>SHELL</b>       | Determine the name of a preferred command interpreter. The default is <i>sh</i> .                     |
| 17743    | <b>TERM</b>        | Determine the name of the terminal type, to indicate in an unspecified manner, if the                 |
| 17744    |                    | internal variable <b>screen</b> is not specified, the number of lines in a screenful of headers. If   |
| 17745    |                    | <b>TERM</b> is not set or is set to null, an unspecified default terminal type will be used and       |
| 17746    |                    | the value of a screenful is unspecified.                                                              |

## VISUAL

Determine a pathname of a utility to invoke when the **visual** command (see **Commands in mailx** on page 492) or **~v** command-escape (see **Command Escapes in mailx** on page 499) is used. If this variable is null or not set, the full-screen editor will be **vi**.

## ASYNCHRONOUS EVENTS

When *mailx* is in Send Mode and standard input is not a terminal, it takes the standard action for all signals.

In Receive Mode, or in Send Mode when standard input is a terminal, if a SIGINT signal is received:

1. If in command mode, the current command, if there is one, will be aborted, and a command-mode prompt will be written.
2. If in input mode:
  - a. If **ignore** is set, *mailx* will write **@\n**, discard the current input line, and continue processing, bypassing the message-abort mechanism described in item 2b.
  - b. If the interrupt was received while sending mail, either when in Receive Mode or in Send Mode, a message will be written, and another subsequent interrupt, with no other intervening characters typed, will be required to abort the mail message. If in Receive Mode and another interrupt is received, a command-mode prompt will be written. If in Send Mode and another interrupt is received, *mailx* will terminate with a non-zero status.

In both cases listed in item b, if the message is not empty:

- i. If **save** is enabled and the file named by *DEAD* can be created, the message will be written to the file named by *DEAD*. If the file exists, the message will be written to replace the contents of the file.
- ii. If **save** is not enabled, or the file named by *DEAD* cannot be created, the message will not be saved.

The *mailx* utility takes the standard action for all other signals.

## STDOUT

In command and input modes, all output, including prompts and messages, is written to standard output.

## STDERR

Used only for diagnostic messages.

## OUTPUT FILES

Various *mailx* commands and command escapes can create or add to files, including the **mbox**, the dead-letter file and secondary mailboxes. When *mailx* is used as described in this specification, these files will be text files, formatted as follows:

```
line beginning with From<space>
[one or more header-lines; see Commands in mailx on page 492]
empty line
[zero or more body lines
empty line]
[line beginning with From<space>...]
```

where each message begins with the **From<space>** line shown, preceded by the beginning of the file or an empty line. (The **From<space>** line is considered to be part of the message header, but

not one of the header-lines referred to in **Commands in mailx** on page 492; thus, it is not affected by the **discard**, **ignore** or **retain** commands.) The formats of the remainder of the **From**<space> line and any additional header lines are unspecified, except that none will be empty. The format of a message body line is also unspecified, except that no line following an empty line can start with **From**<space>; *mailx* will modify any such user-entered message body lines (following an empty line and beginning with **From**<space>) by adding one or more characters to precede the F; it may add these characters to **From**<space> lines that are not preceded by an empty line.

When a message from the system mailbox or entered by the user is not a text file, it is implementation-dependent how such a message is stored in files written by *mailx*.

#### 17801 EXTENDED DESCRIPTION

The *mailx* utility cannot guarantee support for all character encodings in all circumstances. For example, inter-system mail may be restricted to 7-bit data by the underlying network, 8-bit data need not be portable to non-internationalised systems, and so on. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used.

When *mailx* is invoked using one of the Receive Mode synopsis forms, it will write the page of header-summary lines (see below) containing the first new message (if **-N** is not specified), or the first unread message if there are no new messages, or the first message if there are no new or unread messages, followed by a prompt indicating *mailx* can accept regular commands (see **Commands in mailx** on page 492); this is termed *command mode*. When *mailx* is invoked using the Send Mode synopsis and standard input is a terminal, if no subject is specified on the command line and the **asksub** variable is set, a prompt for the subject will be written. At this point *mailx* is in *input mode*. This input mode is also entered when using one of the Receive Mode synopsis forms and a reply or new message is composed using the **reply**, **Reply**, **followup**, **Followup** or **mail** commands. When the message is typed and the end of message is encountered, the message will be passed to the mail delivery software. Commands can be entered by beginning a line with the escape character (by default, tilde (~)) followed by a single command letter and optional arguments. See **Command Escapes in mailx** on page 499 for a summary of these commands.

**Note:** For notational convenience, this section uses the default escape character, tilde, in all references and examples.

At any time, the behaviour of *mailx* is governed by a set of environmental and internal variables. These are flags and valued parameters that can be set and cleared via the *mailx* **set** and **unset** commands.

Regular commands are of the form:

```
[command] [msglist] [argument . . .]
```

If no *command* is specified in command mode, **print** is assumed. In input mode, commands are recognised by the escape character, and lines not treated as commands are taken as input for the message.

In command mode, each message will be assigned a sequential number, starting with 1.

All messages have a state that affects how they are displayed in the header summary and how they are retained or deleted upon termination of *mailx*. There is at any time the notion of a *current* message, marked by a ">" at the beginning of a line in the header summary. All messages are in one of the following states:

*new* The message is present in the system mailbox and has not been viewed by the user or moved to any other state. Messages in state *new* when *mailx* quits will be retained in the system mailbox.

|       |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17839 | <i>unread</i>    | The message has been present in the system mailbox for more than one invocation of <i>mailx</i> and has not been viewed by the user or moved to any other state. Messages in state <i>unread</i> when <i>mailx</i> quits will be retained in the system mailbox.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 17840 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17841 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17842 | <i>read</i>      | The message has been processed by one of the following commands: <b>~f</b> , <b>~m</b> , <b>~F</b> , <b>~M</b> , <b>copy</b> , <b>mbox</b> , <b>next</b> , <b>pipe</b> , <b>print</b> , <b>Print</b> , <b>top</b> , <b>type</b> , <b>Type</b> , <b>undelete</b> . The <b>delete</b> , <b>dp</b> and <b>dt</b> commands may also cause the next message to be marked as <i>read</i> , depending on the value of the <b>autoprint</b> variable. Messages that are in the system mailbox and in state <i>read</i> when <i>mailx</i> quits will be saved in the <b>mbox</b> , unless the internal variable <b>hold</b> was set. Messages that are in the <b>mbox</b> or in a secondary mailbox and in state <i>read</i> when <i>mailx</i> quits will be retained in their current location. |
| 17843 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17844 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17845 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17846 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17847 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17848 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17849 | <i>deleted</i>   | The message has been processed by one of the following commands: <b>delete</b> , <b>dp</b> , <b>dt</b> . A message processed by <b>save</b> will be in state <i>deleted</i> unless the internal variable <b>keepsave</b> was set. Messages in state <i>deleted</i> when <i>mailx</i> quits will be deleted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 17850 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17851 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17852 | <i>preserved</i> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17853 |                  | The message has been processed by a <b>preserve</b> command. When <i>mailx</i> quits, the message will be retained in its current location.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 17854 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17855 |                  | The header-summary line for each message will indicate the state of the message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 17856 |                  | Many commands take an optional list of messages ( <i>msglist</i> ) on which to operate, which defaults to the current message. A <i>msglist</i> is a list of message specifications separated by blank characters, which can include:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 17857 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17858 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17859 | <i>n</i>         | Message number <i>n</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 17860 | <b>+</b>         | The next undeleted message, or the next deleted message for the <b>undelete</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 17861 | <b>-</b>         | The next previous undeleted message, or the next previous deleted message for the <b>undelete</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 17862 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17863 | <b>.</b>         | The current message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 17864 | <b>^</b>         | The first undeleted message, or the first deleted message for the <b>undelete</b> command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 17865 | <b>\$</b>        | The last message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 17866 | <b>*</b>         | All messages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 17867 | <i>n-m</i>       | An inclusive range of message numbers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 17868 | <i>address</i>   | All messages from <i>address</i> ; any address as shown in a header summary will be matchable in this form.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 17869 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17870 | <i>/string</i>   | All messages with <i>string</i> in the subject line (case ignored).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 17871 | <i>:c</i>        | All messages of type <i>c</i> , where <i>c</i> must be one of:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17872 | <b>d</b>         | deleted messages                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 17873 | <b>n</b>         | new messages                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 17874 | <b>o</b>         | old messages (any not in state <i>read</i> or <i>new</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 17875 | <b>r</b>         | read messages                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 17876 | <b>u</b>         | unread messages                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17877 |                  | Other commands take an optional message ( <i>message</i> ) on which to operate, which defaults to the current message. All of the forms allowed for <i>msglist</i> are also allowed for <i>message</i> , but if more than one message is specified, only the first will be operated on.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 17878 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 17879 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |



17880 Other arguments are usually arbitrary strings whose usage depends on the command involved.

### 17881 **Start-up in mailx**

17882 At start-up time, *mailx* will take the following steps in sequence:

- 17883 1. Establish all variables at their stated default values.
- 17884 2. Process command-line options, overriding corresponding default values.
- 17885 3. Import any of the *DEAD*, *EDITOR*, *MBOX*, *LISTER*, *PAGER*, *SHELL* or *VISUAL* variables  
17886 that are present in the environment, overriding the corresponding default values.
- 17887 4. Read *mailx* commands from an unspecified system start-up file, unless the **-n** option is  
17888 given, to initialise any internal *mailx* variables and aliases.
- 17889 5. Process the start-up file of *mailx* commands named in the user *MAILRC* variable.

17890 Most regular *mailx* commands are valid inside start-up files, the most common use being to set  
17891 up initial display options and alias lists. The following commands are invalid in the start-up file:  
17892 EX **!, edit, hold, mail, preserve, reply, Reply, shell, visual, Copy, followup and Followup.** Any  
17893 errors in the start-up file will either cause *mailx* to terminate with a diagnostic message and a  
17894 non-zero status or to continue after writing a diagnostic message, ignoring the remainder of the  
17895 lines in the start-up file.

17896 A blank line in a start-up file is ignored.

### 17897 **Internal Variables in mailx**

17898 The following variables are internal *mailx* variables. Each internal variable can be set via the  
17899 *mailx set* command at any time. The **unset** and **set no name** commands can be used to erase  
17900 variables.

17901 In the following list, variables shown as:

17902       variable

17903 represent Boolean values. Variables shown as:

17904       variable=value

17905 will be assigned string or numeric values. For string values, the rules in **Commands in mailx** on  
17906 page 492 concerning filenames and quoting also apply.

17907 The defaults specified here may be changed by the implementation-dependent system start-up  
17908 file unless the user specifies the **-n** option.

17909 EX **allnet** All network names whose login name components match are treated as identical. This  
17910 causes the *msglist* message specifications to behave similarly. The default is **noallnet**.  
17911 See also the **alternates** command and the **metoo** variable.

17912 **append** Append messages to the end of the **mbox** file upon termination instead of placing them  
17913 at the beginning. The default is **noappend**. This variable will not affect the **save**  
17914 command when saving to the **mbox**.

17915 **ask**

17916 **asksub** Prompt for a subject line on outgoing mail if one is not specified on the command line  
17917 with the **-s** option. The **ask** and **asksub** forms are synonyms; the system will refer to  
17918 **asksub** and **noasksub** in its messages, but will accept **ask** and **noask** as user input to  
17919 mean **asksub** and **noasksub**. It is not possible to set both **ask** and **noasksub**, or **noask**  
17920 and **asksub**. The default is **asksub**, but no prompting will be done if standard input is  
17921 not a terminal.

|          |                            |                                                                                                                |
|----------|----------------------------|----------------------------------------------------------------------------------------------------------------|
| 17922    | <b>askbcc</b>              | Prompt for the blind copy list. The default is <b>noaskbcc</b> .                                               |
| 17923    | <b>askcc</b>               | Prompt for the copy list. The default is <b>noaskcc</b> .                                                      |
| 17924    | <b>autoprint</b>           |                                                                                                                |
| 17925    |                            | Enable automatic writing of messages after <b>delete</b> and <b>undelete</b> commands. The                     |
| 17926    |                            | default is <b>noautoprint</b> .                                                                                |
| 17927    | <b>bang</b>                | Enable the special-case treatment of exclamation-marks (!) in escape command lines;                            |
| 17928    |                            | see the <b>escape</b> command and <b>Command Escapes in mailx</b> on page 499. The default is                  |
| 17929    |                            | <b>nobang</b> , disabling the expansion of "!" in the <i>command</i> argument to the ~! command                |
| 17930    |                            | and the ~<!command escape.                                                                                     |
| 17931    | <b>cmd=command</b>         |                                                                                                                |
| 17932    |                            | Set the default command to be invoked by the <b>pipe</b> command. The default is <b>nocmd</b> .                |
| 17933    | <b>crt=number</b>          |                                                                                                                |
| 17934    |                            | Pipe messages having more than <i>number</i> lines through the command specified by the                        |
| 17935    |                            | value of the <i>PAGER</i> variable. The default is <b>nocrt</b> . If it is set to null, the value used is      |
| 17936    |                            | implementation-dependent.                                                                                      |
| 17937 EX | <b>debug</b>               | Enable verbose diagnostics for debugging. Messages are not delivered. The default is                           |
| 17938    |                            | <b>nodebug</b> .                                                                                               |
| 17939    | <b>dot</b>                 | When <b>dot</b> is set, a period on a line by itself during message input from a terminal also                 |
| 17940    |                            | signifies end-of-file (in addition to normal end-of-file). The default is <b>nodot</b> . If                    |
| 17941    |                            | <b>ignoreeof</b> is set (see below), a setting of <b>nodot</b> will be ignored and the period is the only      |
| 17942    |                            | method to terminate input mode.                                                                                |
| 17943    | <b>escape=c</b>            |                                                                                                                |
| 17944    |                            | Set the command escape character to be the character <i>c</i> . By default, the command                        |
| 17945    |                            | escape character is tilde. If <b>escape</b> is unset, tilde will be used; if it is set to null,                |
| 17946    |                            | command escaping will be disabled.                                                                             |
| 17947    | <b>flipr</b>               | Reverse the meanings of the <b>R</b> and <b>r</b> commands. The default is <b>noflipr</b> .                    |
| 17948    | <b>folder=directory</b>    |                                                                                                                |
| 17949    |                            | The default directory for saving mail files. User-specified filenames beginning with a                         |
| 17950    |                            | plus sign (+) will be expanded by preceding the filename with this directory name to                           |
| 17951    |                            | obtain the real pathname. If <i>directory</i> does not start with a slash (/), the contents of                 |
| 17952    |                            | <i>HOME</i> will be prefixed to it. The default is <b>nofolder</b> . If <b>folder</b> is unset or set to null, |
| 17953    |                            | user-specified filenames beginning with "+" refer to files in the current directory that                       |
| 17954    |                            | begin with the literal "+" character. See also <b>outfolder</b> below. The <b>folder</b> value need            |
| 17955    |                            | not affect the processing of the files named in <i>MBOX</i> and <i>DEAD</i> .                                  |
| 17956    | <b>header</b>              | Enable writing of the header summary when entering <i>mailx</i> in Receive Mode. The                           |
| 17957    |                            | default is <b>header</b> .                                                                                     |
| 17958    | <b>hold</b>                | Preserve all messages that are read in the system mailbox instead of putting them in the                       |
| 17959    |                            | <b>mbox</b> save file. The default is <b>nohold</b> .                                                          |
| 17960    | <b>ignore</b>              | Ignore interrupts while entering messages. The default is <b>noignore</b> .                                    |
| 17961    | <b>ignoreeof</b>           |                                                                                                                |
| 17962    |                            | Ignore normal end-of-file during message input. Input can be terminated only by                                |
| 17963    |                            | entering a period (.) on a line by itself or by the ~. command escape. The default is                          |
| 17964    |                            | <b>noignoreeof</b> . See also <b>dot</b> above.                                                                |
| 17965    | <b>indentprefix=string</b> |                                                                                                                |
| 17966    |                            | A string that will be prefixed to each line that is inserted into the message by the ~m                        |

|          |                               |                                                                                                             |
|----------|-------------------------------|-------------------------------------------------------------------------------------------------------------|
| 17967    |                               | command escape. This variable defaults to one tab character.                                                |
| 17968    | <b>keep</b>                   | When a system mailbox, secondary mailbox or <b>mbox</b> is empty, truncate it to zero                       |
| 17969    |                               | length instead of removing it. The default is <b>nokeep</b> .                                               |
| 17970    | <b>keepsave</b>               |                                                                                                             |
| 17971    |                               | Keep messages that have been saved in other files in the system mailbox instead of                          |
| 17972    |                               | deleting them. The default is <b>nokeepsave</b> .                                                           |
| 17973    | <b>metoo</b>                  | Suppress the deletion of the login name of the user from the recipient list when                            |
| 17974    |                               | replying to a message or sending to a group. The default is <b>nometoo</b> .                                |
| 17975 EX | <b>onehop</b>                 | When responding to a message that was originally sent to several recipients, the other                      |
| 17976    |                               | recipient addresses are normally forced to be relative to the originating author's                          |
| 17977    |                               | machine for the response. This flag disables alteration of the recipients' addresses,                       |
| 17978    |                               | improving efficiency in a network where all machines can send directly to all other                         |
| 17979    |                               | machines (that is, one hop away). The default is <b>noonehop</b> .                                          |
| 17980    | <b>outfolder</b>              |                                                                                                             |
| 17981    |                               | Cause the files used to record outgoing messages to be located in the directory                             |
| 17982    |                               | specified by the <b>folder</b> variable unless the pathname is absolute. The default is                     |
| 17983    |                               | <b>nooutfolder</b> . See the <b>record</b> variable.                                                        |
| 17984    | <b>page</b>                   | Insert a form-feed after each message sent through the pipe created by the <b>pipe</b>                      |
| 17985    |                               | command. The default is <b>nopage</b> .                                                                     |
| 17986    | <b>prompt=string</b>          |                                                                                                             |
| 17987    |                               | Set the command-mode prompt to <i>string</i> . If <i>string</i> is null or if <b>noprompt</b> is set, no    |
| 17988    |                               | prompting will occur. The default is to prompt with the string "? ".                                        |
| 17989    | <b>quiet</b>                  | Refrain from writing the opening message and version when entering <i>mailx</i> . The                       |
| 17990    |                               | default is <b>noquiet</b> .                                                                                 |
| 17991    | <b>record=file</b>            |                                                                                                             |
| 17992    |                               | Record all outgoing mail in the file with the pathname <i>file</i> . The default is <b>norecord</b> . See   |
| 17993    |                               | also <b>outfolder</b> above.                                                                                |
| 17994    | <b>save</b>                   | Enable saving of messages in the dead-letter file on interrupt or delivery error. See the                   |
| 17995    |                               | variable <i>DEAD</i> for the location of the dead-letter file. The default is <b>save</b> .                 |
| 17996    | <b>screen=number</b>          |                                                                                                             |
| 17997    |                               | Set the number of lines in a screenful of headers for the <b>headers</b> and <b>z</b> commands. If          |
| 17998    |                               | <b>screen</b> is not specified, a value based on the terminal type identified by the <i>TERM</i>            |
| 17999    |                               | environment variable, the window size, the baud rate, or some combination of these                          |
| 18000    |                               | will be used.                                                                                               |
| 18001 EX | <b>sendmail=shell-command</b> |                                                                                                             |
| 18002    |                               | Alternative command for delivering messages. The default is implementation-                                 |
| 18003    |                               | dependent. ( <b>LEGACY</b> )                                                                                |
| 18004 EX | <b>sendwait</b>               |                                                                                                             |
| 18005    |                               | Wait for the background mailer to finish before returning. The default is <b>nosendwait</b> .               |
| 18006    | <b>showto</b>                 | When the sender of the message was the user who is invoking <i>mailx</i> , write the                        |
| 18007    |                               | information from the To: line instead of the From: line in the header summary. The                          |
| 18008    |                               | default is <b>noshowto</b> .                                                                                |
| 18009    | <b>sign=string</b>            |                                                                                                             |
| 18010    |                               | Set the variable inserted into the text of a message when the <b>~a</b> command escape is                   |
| 18011    |                               | given. The default is <b>nosign</b> . The character sequences <b>\t</b> and <b>\n</b> are recognised in the |

18012 variable as tab and newline characters, respectively. (See also `~i` in **Command Escapes**  
 18013 **in mailx** on page 499.)

18014 **Sign=string**  
 18015 Set the variable inserted into the text of a message when the `~A` command escape is  
 18016 given. The default is `noSign`. The character sequences `\t` and `\n` will be recognised in  
 18017 the variable as tab and newline characters, respectively.

18018 **toplines=number**  
 18019 Set the number of lines of the message to write with the **top** command. The default is 5.

## 18020 **Commands in mailx**

18021 The following *mailx* commands are provided. In the following list, header refers to lines from  
 18022 the message header, as shown in the OUTPUT FILES section. Header-line refers to lines within  
 18023 the header that begin with one or more non-white-space characters, immediately followed by a  
 18024 colon and white space and continuing until the next line beginning with a non-white-space  
 18025 character or an empty line. Header-field refers to the portion of a header line prior to the first  
 18026 colon in that line.

18027 For each of the commands listed below, the command can be entered as the abbreviation (those  
 18028 characters in the Synopsis command word preceding the `]`), the full command (all characters  
 18029 shown for the command word, omitting the `[` and `]`), or any truncation of the full command  
 18030 down to the abbreviation. For example, the **exit** command (shown as **ex[it]** in the Synopsis)  
 18031 can be entered as **ex**, **exi** or **exit**.

18032 The arguments to commands can be quoted, using the following methods:

- 18033 • An argument can be enclosed between paired double-quotes (" ") or single-quotes ( ' '); any  
 18034 white space, shell word expansion or backslash characters within the quotes will be treated  
 18035 literally as part of the argument. A double-quote will be treated literally within single-quotes  
 18036 and *vice versa*. These special properties of the quote marks occur only when they are paired  
 18037 at the beginning and end of the argument.
- 18038 • A backslash outside of the enclosing quotes is discarded and the following character treated  
 18039 literally as part of the argument.
- 18040 • An unquoted backslash at the end of a command line is discarded and the next line continues  
 18041 the command.

18042 Filenames, where expected, are subjected to the process of shell word expansions (see Section 2.6  
 18043 on page 31); if more than a single pathname results and the command is expecting one file, the  
 18044 effects are unspecified. If the filename begins with an unquoted plus sign, it will not be  
 18045 expanded, but treated as the named file (less the leading plus) in the **folder** directory. (See the  
 18046 **folder** variable.)

## 18047 **Declare Aliases**

18048 *Synopsis:*    a[lias] [*alias* [*address...*]]  
 18049 *Synopsis:*    g[roup] [*alias* [*address...*]]

18050 Add the given addresses to the alias specified by *alias*. The names will be substituted when *alias*  
 18051 is used as a recipient address specified by the user in an outgoing message (that is, other  
 18052 recipients addressed indirectly through the **reply** command will not be substituted in this  
 18053 manner). Mail address alias substitution applies only when the alias string is used as a full  
 18054 address; for example, when **hlj** is an alias, **hlj@posix.com** does not trigger the alias substitution.  
 18055 If no arguments are given, write a listing of the current aliases to standard output. If only an  
 18056 *alias* argument is given, write a listing of the specified alias to standard output. These listings

18057 need not reflect the same order of addresses that were entered.

## 18058 **Declare Alternatives**

18059 *Synopsis:* alt[ernates] *name...*

18060 Declare a list of alternative names for the user's login. When responding to a message, these  
18061 names will be removed from the list of recipients for the response. The comparison of names  
18062 will be in a case-insensitive manner. With no arguments, **alternates** will write the current list of  
18063 alternative names.

## 18064 **Change Current Directory**

18065 *Synopsis:* cd [*directory*]

18066 *Synopsis:* ch[dir] [*directory*]

18067 Change directory. If *directory* is not specified, the contents of *HOME* will be used.

## 18068 **Copy Messages**

18069 *Synopsis:* c[opy] [*file*]

18070 *Synopsis:* c[opy] [*msglist*] *file*

18071 EX *Synopsis:* C[opy] [*msglist*]

18072 Copy messages to the file named by the pathname *file* without marking the messages as saved.  
18073 Otherwise, it is equivalent to the **save** command.

18074 EX In the capitalised form, save the specified messages in a file whose name is derived from the  
18075 author of the message to be saved, without marking the messages as saved. Otherwise, it is  
18076 equivalent to the **Save** command.

## 18077 **Delete Messages**

18078 *Synopsis:* d[ele]te [*msglist*]

18079 Mark messages for deletion from the mailbox. The deletions will not occur until *mailx* quits (see  
18080 the **quit** command) or changes mailboxes (see the **folder** command). If **autoprint** is set, the next  
18081 message after the last one deleted will be written; if there is no subsequent message, the  
18082 previous message, if it exists, will be written. In the case of no subsequent or previous message,  
18083 or when **noautoprint** is set, the *mailx* prompt will be written.

## 18084 **Discard Header Fields**

18085 *Synopsis:* di[scard] [*header-field...*]

18086 *Synopsis:* ig[nore] [*header-field...*]

18087 Suppress the specified header fields when writing messages. Specified *header-fields* will be  
18088 added to the list of suppressed header fields. Examples of header fields to ignore are **status** and  
18089 **cc**. The fields will be included when the message is saved. The **Print** and **Type** commands  
18090 override this command. The comparison of header fields is in a case-insensitive manner. If no  
18091 arguments are specified, write a list of the currently suppressed header fields to standard  
18092 output; the listing need not reflect the same order of header fields that were entered.

18093 If both **retain** and **discard** commands are given, **discard** commands are ignored.

18094 **Delete Messages and Display**18095 *Synopsis:* dp [*msglist*]18096 *Synopsis:* dt [*msglist*]

18097 Delete the specified messages from the mailbox and write the next message after the last one  
 18098 deleted. If there is no subsequent message, the *mailx* prompt will be written.

18099 **Echo a String**18100 EX *Synopsis:* ec[ho] *string* . . .18101 Echo the given strings, equivalent to the shell *echo* utility.18102 **Edit Messages**18103 *Synopsis:* e[dit] [*msglist*]

18104 Edit the given messages. The messages will be placed in a temporary file and the utility named  
 18105 by the *EDITOR* variable will be invoked to edit the file.

18106 The **edit** command merely edits the specified messages in a temporary file. It does not modify  
 18107 the contents of those messages in the mailbox.

18108 **Exit**18109 *Synopsis:* ex[it]18110 *Synopsis:* x[it]

18111 Exit from *mailx* without changing the mailbox. No messages will be saved in the **mbox** (see also  
 18112 **quit**).

18113 **Change Folder**18114 *Synopsis:* fi[le] [*file*]18115 *Synopsis:* fold[er] [*file*]

18116 Quit (see the **quit** command) from the current file of messages and read in the file named by the  
 18117 pathname *file*. If no argument is given, the name and status of the current mailbox will be  
 18118 written.

18119 Several unquoted special characters are recognised when used as *file* names, with the following  
 18120 substitutions:

18121 % The system mailbox for the invoking user.

18122 %*user* The system mailbox for *user*.

18123 # The previous file.

18124 & The current **mbox**.18125 +*file* The named file in the **folder** directory. (See the **folder** variable.)

18126 The default file is the current mailbox.

18127 **Display List of Folders**18128 *Synopsis:* folders18129 Write the names of the files in the directory set by the **folder** variable.

18130 **Follow up Specified Messages**18131 EX *Synopsis:* fo[llowup] [*message*]18132 *Synopsis:* F[ollowup] [*msglist*]

18133 In the lower-case form, respond to a message, recording the response in a file whose name is  
 18134 derived from the author of the message. Overrides the **record** variable, if set. See also the **save**  
 18135 and **copy** commands and **outfolder**.

18136 In the capitalised form, respond to the first message in the *msglist*, sending the message to the  
 18137 author of each message in the *msglist*. The subject line is taken from the first message and the  
 18138 response is recorded in a file whose name is derived from the author of the first message. See  
 18139 also the **Save** and **Copy** commands and **outfolder**.

18140 **Display Header Summary for Specified Messages**18141 *Synopsis:* f[rom] [*msglist*]

18142 Write the header summary for the specified messages.

18143 **Display Header Summary**18144 *Synopsis:* h[eaders] [*message*]

18145 Write the page of headers that includes the message specified. The **screen** variable sets the  
 18146 number of headers per page. See also the **z** command.

18147 **Help**18148 *Synopsis:* hel[p]18149 *Synopsis:* ?

18150 Write a summary of commands.

18151 **Hold Messages**18152 *Synopsis:* ho[ld] [*msglist*]18153 *Synopsis:* pre[serve] [*msglist*]

18154 Mark the messages in *msglist* to be retained in the mailbox when *mailx* terminates. This  
 18155 overrides any commands that might previously have marked the messages to be deleted.  
 18156 During the current invocation of *mailx*, only the **delete**, **dp** or **dt** commands will remove the  
 18157 *preserve* marking of a message.

18158 **Execute Commands Conditionally**18159 *Synopsis:* i[f] s|r18160 *mail-commands*

18161 el[se]

18162 *mail-commands*

18163 en[dif]

18164 Execute commands conditionally, where **if s** will execute the following *mail-commands*, up to an  
 18165 **else** or **endif**, if the program is in Send Mode, and **if r** will cause the *mail-commands* to be  
 18166 executed only in Receive Mode.

**List Available Commands**

*Synopsis:*    l[ist]

Write a list of all commands available. No explanation is given.

**Mail a Message**

*Synopsis:*    m[ail] *address...*

Mail a message to the specified addresses or aliases.

**Direct Messages to mbox**

*Synopsis:*    mb[ox] [*msglist*]

Arrange for the given messages to end up in the **mbox** save file when *mailx* terminates normally. See **MBOX**. See also the **exit** and **quit** commands.

**Process Next Specified Message**

*Synopsis:*    n[ext] [*message*]

Go to the next message matching *message*.

**Pipe Message**

*Synopsis:*    pi[pe] [[*msglist*] *command*]

*Synopsis:*    | [[*msglist*] *command*]

Pipe the messages through the given *command* by invoking the command interpreter specified by *SHELL* with two arguments: **-c** and *command*. (See also *sh -c*.) The command must be given as a single argument. Quoting, described previously, can be used to accomplish this. If no arguments are given, the current message will be piped through the command specified by the value of the **cmd** variable. If the **page** variable is set, a form-feed character will be inserted after each message.

**Display Message with Headers**

*Synopsis:*    P[rint] [*msglist*]

*Synopsis:*    T[ype] [*msglist*]

Write the specified messages, including all header lines, to standard output. Override suppression of lines by the **discard**, **ignore** and **retain** commands. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable will be paged through the command specified by the **PAGER** environment variable.

**Display Message**

*Synopsis:*    p[rint] [*msglist*]

*Synopsis:*    t[ype] [*msglist*]

Write the specified messages to standard output. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable will be paged through the command specified by the **PAGER** environment variable.



18202      **Quit**18203      *Synopsis:*    q[uit]18204      *Synopsis:*    end-of-file

18205      Terminate *mailx*, storing messages that were read in **mbox** (if the current mailbox is the system  
 18206      mailbox and unless **hold** is set), deleting messages that have been explicitly saved (unless  
 18207      **keepsave** is set), discarding messages that have been deleted and saving all remaining messages  
 18208      in the mailbox.

18209      **Reply to a Message List**18210      *Synopsis:*    R[eply] [msglist]18211      *Synopsis:*    R[espond] [msglist]

18212      Mail a reply message to the sender of each message in the *msglist*. The subject line will be  
 18213      formed by concatenating **Re:**<space> (unless it already begins with that string) and the subject  
 18214      from the first message. If **record** is set to a filename, the response will be saved at the end of that  
 18215      file.

18216      See also the **flpr** variable.18217      **Reply to a Message**18218      *Synopsis:*    r[eply] [message]18219      *Synopsis:*    r[espond] [message]

18220      Mail a reply message to all recipients included in the header of the message. The subject line  
 18221      will be formed by concatenating **Re:**<space> (unless it already begins with that string) and the  
 18222      subject from the message. If **record** is set to a filename, the response will be saved at the end of  
 18223      that file.

18224      See also the **flpr** variable.18225      **Retain Header Fields**18226      *Synopsis:*    ret[ain] [header-field...]

18227      Retain the specified header fields when writing messages. This command will override all  
 18228      **discard** and **ignore** commands. The comparison of header fields is in a case-insensitive manner.  
 18229      If no arguments are specified, write a list of the currently retained header fields to standard  
 18230      output; the listing need not reflect the same order of header fields that were entered.

18231      **Save Messages**18232      *Synopsis:*    s[ave] [file]18233      *Synopsis:*    s[ave] [msglist] file18234 EX      *Synopsis:*    S[ave] [msglist]

18235      Save the specified messages in the file named by the pathname *file*, or the **mbox** if the *file*  
 18236      argument is omitted. The file will be created if it does not exist; otherwise, the messages will be  
 18237      appended to the file. The message will be deleted from the mailbox when *mailx* terminates  
 18238      unless **keepsave** is set.

18239 EX      In the capitalised form, save the specified messages in a file whose name is derived from the  
 18240      author of the first message. The name of the file is taken to be the author's name with all  
 18241      network addressing stripped off. See also the Copy, **followup** and Followup commands and  
 18242      **outfolder** variable.

**Set Variables**

*Synopsis:*    se[t] [name=[*string*]]... [name=*number*...] [noname...]

Define one or more variables called *name*. The variable can be given a null, string or numeric value. Quoting and backslash escapes can occur anywhere in *string*, as described previously, as if the *string* portion of the argument were the entire argument. The forms *name* and *name=* are equivalent to *name=""* for variables that take string values. The **set** command without arguments will write a list of all defined variables and their values. The *noname* form is equivalent to **unset** *name*.

**Invoke a Shell**

*Synopsis:*    sh[ell]

Invoke an interactive command interpreter (see also *SHELL*).

**Display Message Size**

*Synopsis:*    si[ze] [*msglist*]

Write the size in bytes of each of the specified messages.

**Read mailx Commands From a File**

*Synopsis:*    so[urce] *file*

Read and execute commands from the file named by the pathname *file* and return to command mode.

**Display Beginning of Messages**

*Synopsis:*    to[p] [*msglist*]

Write the top few lines of each of the specified messages. If the **toplines** variable is set, it is taken as the number of lines to write. The default is 5.

**Touch Messages**

*Synopsis:*    tou[ch] [*msglist*]

Touch the specified messages. If any message in *msglist* is not specifically deleted nor saved in a file, it will be placed in the **mbox** upon normal termination. See **exit** and **quit**.

**Delete Aliases**

*Synopsis:*    una[lias] [*alias*]...

Delete the specified alias names. If a specified alias does not exist, the results are unspecified.

**Undelete Messages**

*Synopsis:*    u[ndelete] [*msglist*]

Remove the deleted markings from the specified messages. If **autoprint** is set, the last message of those restored will be written. If *msglist* is not specified, it defaults to the first deleted message following the current message that has not been undeleted if there is one, or the last deleted message preceding the current message that has not been undeleted otherwise.

18278      **Unset Variables**18279      *Synopsis:*      `uns[et] name...`

18280      Cause the specified variables to be erased.

18281      **Edit Message with Full-screen Editor**18282      *Synopsis:*      `v[isual] [msglist]`18283      Edit the given messages with a screen editor. The messages are placed in a temporary file, and  
18284      the utility named by the *VISUAL* variable will be invoked to edit the file. The default editor is *vi*.18285      The **visual** command merely edits the specified messages in a temporary file. It does not modify  
18286      the contents of those messages in the mailbox.18287      **Write Messages to a File**18288      *Synopsis:*      `w[rite] [msglist] file`18289      Write the given messages to the file specified by the pathname *file*, minus the message header.  
18290      Otherwise, it is equivalent to the **save** command.18291      **Scroll Header Display**18292      *Synopsis:*      `z[+|-]`18293      Scroll the header display forward (if "+" is specified or if no option is specified) or backward (if  
18294      "-" is specified) one screenful. The number of headers written is set by the **screen** variable.18295      **Invoke Shell Command**18296      *Synopsis:*      `!command`18297      Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and *command*.  
18298      (See also *sh -c*.) If the **bang** variable is set, each unescaped occurrence of "!" in *command* is  
18299      replaced with the command executed by the previous "!" command or ~! command escape.18300      **Null Command**18301      *Synopsis:*      `# comment`18302      This null command (comment) will be ignored by *mailx*.18303      **Display Current Message Number**18304      *Synopsis:*      `=`

18305      Write the current message number.

18306      **Command Escapes in mailx**18307      The following commands can be entered only from input mode, by beginning a line with the  
18308      escape character (by default, tilde (~)). See the **escape** variable description for changing this  
18309      special character. The format for the commands is:18310      `<ESC><command-char><separator>[<arguments>]`18311      where the *<separator>* can be zero or more blank characters.18312      In the following descriptions, the argument *command* (but not *mailx-command*) must be a shell  
18313      command string. Any string acceptable to the command interpreter specified by the *SHELL*

18314 variable when it is invoked as `-c command_string` is valid. The command can be presented as  
 18315 multiple arguments (that is, quoting is not required).

18316 Command escapes that are listed with *msglist* or *mailx-command* arguments are invalid in Send  
 18317 Mode and produce unspecified results.

18318 `~! command`  
 18319 Invoke the command interpreter specified by *SHELL* with two arguments: `-c` and  
 18320 *command*; and then return to input mode. If the **bang** variable is set, each unescaped  
 18321 occurrence of `"!"` in *command* is replaced with the command executed by the previous  
 18322 `"!"` command or `~!` command escape.

18323 `~.` Simulate end-of-file (terminate message input).

18324 `~: mailx-command`  
 18325 `~_ mailx-command`  
 18326 Perform the command-level request.

18327 `~?` Write a summary of command escapes.

18328 `~A` This is equivalent to `~i Sign`.

18329 `~a` This is equivalent to `~i sign`.

18330 `~b name...`  
 18331 Add the *names* to the blind carbon copy (Bcc) list.

18332 `~c name...`  
 18333 Add the *names* to the carbon copy (Cc) list.

18334 `~d` Read in the dead-letter file. See *DEAD* for a description of this file.

18335 `~e` Invoke the editor, as specified by the *EDITOR* environment variable, on the partial  
 18336 message.

18337 `~f [msglist]`  
 18338 Forward the specified messages. The specified messages will be inserted into the  
 18339 current message without alteration. This command escape will also insert message  
 18340 headers into the message with field selection affected by the **discard**, **ignore** and **retain**  
 18341 commands.

18342 `~F [msglist]`  
 18343 This will be the equivalent of the `~f` command escape, except that all headers will be  
 18344 included in the message, regardless of previous **discard**, **ignore** and **retain** commands.

18345 `~h` If standard input is a terminal, prompt for a Subject line and the To, Cc and Bcc lists.  
 18346 Other implementation-dependent headers may also be presented for editing. If the  
 18347 field is written with an initial value, it can be edited as if it had just been typed.

18348 `~i string` Insert the value of the named variable, followed by a newline character, into the text of  
 18349 the message. If the string is unset or null, the message will not be changed.

18350 `~m [msglist]`  
 18351 Insert the specified messages into the message, prefixing non-empty lines with the  
 18352 string in the **indentprefix** variable. This command escape will also insert message  
 18353 headers into the message, with field selection affected by the **discard**, **ignore** and **retain**  
 18354 commands.

18355 `~M [msglist]`  
 18356 This will be the equivalent of the `~m` command escape, except that all headers will be  
 18357 included in the message, regardless of previous **discard**, **ignore** and **retain** commands.

18358       ~p       Write the message being entered. If the message is longer than **crt** lines (see **Internal**  
 18359       **Variables in mailx** on page 489), the output will be paginated as described for the  
 18360       **PAGER** variable.

18361       ~q       Quit (see the **quit** command) from input mode by simulating an interrupt. If the body  
 18362       of the message is not empty, the partial message will be saved in the dead-letter file.  
 18363       See **DEAD** for a description of this file.

18364       ~r *file*  
 18365       ~< *file*  
 18366       ~< !*command*  
 18367       Read in the file specified by the pathname *file*. If the argument begins with an  
 18368       exclamation-mark (!), the rest of the string is taken as an arbitrary system command;  
 18369       the command interpreter specified by **SHELL** will be invoked with two arguments: **-c**  
 18370       and *command*. The standard output of *command* will be inserted into the message.

18371       ~s *string* Set the subject line to *string*.

18372       ~t *name...*  
 18373       Add the given *names* to the To list.

18374       ~v       Invoke the full-screen editor, as specified by the **VISUAL** environment variable, on the  
 18375       partial message.

18376       ~w *file* Write the partial message, without the header, onto the file named by the pathname *file*.  
 18377       The file will be created or the message will be appended to it if the file exists.

18378       ~x       Exit as with ~q, except the message will not be saved in the dead-letter file.

18379       ~| *command*  
 18380       Pipe the body of the message through the given *command* by invoking the command  
 18381       interpreter specified by **SHELL** with two arguments: **-c** and *command*. If the *command*  
 18382       returns a successful exit status, the standard output of the command will replace the  
 18383       message. Otherwise the message will remain unchanged. If the *command* fails, an error  
 18384       message giving the exit status will be written.

18385   **EXIT STATUS**  
 18386       When the **-e** option is specified, the following exit values are returned:

18387       0   Mail was found.  
 18388       >0   Mail was not found or an error occurred.

18389       Otherwise, the following exit values are returned:

18390       0   Successful completion; note that this status implies that all messages were *sent*, but it gives  
 18391       no assurances that any of them were actually *delivered*.  
 18392       >0   An error occurred.

18393   **CONSEQUENCES OF ERRORS**  
 18394       When in input mode (Receive Mode) or Send Mode:

- 18395       • If an error is encountered processing a command escape (see **Command Escapes in mailx** on  
 18396       page 499), a diagnostic message will be written to standard error, and the message being  
 18397       composed may be modified, but this condition will not prevent the message from being sent.
- 18398       • Other errors will prevent the sending of the message.

18399       When in command mode:

- 18400       • Default.

18401 **APPLICATION USAGE**

18402       Delivery of messages to remote systems requires the existence of communication paths to such  
18403       systems. These need not exist.

18404       Input lines are limited to {LINE\_MAX} bytes, but mailers between systems may impose more  
18405       severe line-length restrictions. This specification does not place any restrictions on the length of  
18406       messages handled by *mailx*, and for delivery of local messages the only limitations should be the  
18407       normal problems of available disk space for the target mail file. When sending messages to  
18408       external machines, applications are advised to limit messages to less than 50 kilobytes because  
18409       many mail gateways impose message-length restrictions.

18410       The format of the system mailbox is intentionally unspecified. Not all systems will implement  
18411       system mailboxes as flat files, particularly with the advent of multimedia mail messages. Some  
18412       system mailboxes may be multiple files, others records in a database. The internal format of the  
18413       messages themselves are specified with the historical format from Version 7, but only after they  
18414       have been saved in some file other than the system mailbox. This was done so that many  
18415       historical applications expecting text-file mailboxes will not be broken.

18416       Some new formats for messages can be expected in the future, probably including binary data,  
18417       bit maps and various multimedia objects. As described here, *mailx* is not prohibited from  
18418       handling such messages, but it must store them as text files in secondary mailboxes (unless  
18419       some extension, such as a variable or command-line option, is used to change the stored format).  
18420       Its method of doing so is implementation-dependent and might include translating the data into  
18421       text-file-compatible or readable form or omitting certain portions of the message from the stored  
18422       output.

18423       The **discard** and **ignore** commands are not inverses of the **retain** command. The **retain**  
18424       command discards all header-fields except those explicitly retained. The **discard** command  
18425       keeps all header-fields except those explicitly discarded. If headers exist on the retained header  
18426       list, **discard** and **ignore** commands are ignored.

18427 **EXAMPLES**

18428       None.

18429 **FUTURE DIRECTIONS**

18430       The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
18431       interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
18432       corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
18433       finalised.

18434 **SEE ALSO**

18435       *ed, ls, mail, more, vi.*

18436 **CHANGE HISTORY**

18437       First released in Issue 2.

18438 **Issue 4**

18439       Aligned with the ISO/IEC 9945-2: 1993 standard.

18440       This utility is now mandatory; it is optional in Issue 3.

18441 **Issue 5**

18442       The description of the EDITOR environment variable is changed to indicate that *ed* is the default  
18443       editor if this variable is not set. In previous issues, this default was not stated explicitly at this  
18444       point but was implied further down in the text.

18445       FUTURE DIRECTIONS section added.

18446 **NAME**18447       make — maintain, update and regenerate groups of programs (**DEVELOPMENT**)18448 **SYNOPSIS**18449       make [-einpqrst][-f *makefile*...] [-k | -S][*macros=name*]...  
18450       [*target\_name*...]18451 **DESCRIPTION**18452       The *make* utility can be used as a part of software development to update files that are derived  
18453       from other files. A typical case is one where object files are derived from the corresponding  
18454       source files. The *make* utility examines time relationships and updates those derived files (called  
18455       targets) that have modified times earlier than the modified times of the files (called  
18456       prerequisites) from which they are derived. A description file (makefile) contains a description  
18457       of the relationships between files, and the commands that must be executed to update the  
18458       targets to reflect changes in their prerequisites. Each specification, or rule, consists of a target,  
18459       optional prerequisites and optional commands to be executed when a prerequisite is newer than  
18460       the target. There are two types of rule:

- 18461
- inference rules, which have one target name with at least one period (.) and no slash (/)
  - target rules, which can have more than one target name.

18463       In addition, *make* has a collection of built-in macros and inference rules that infer prerequisite  
18464       relationships to simplify maintenance of programs.

18465       To receive exactly the behaviour described in this section, a portable makefile must:

- 18466
- include the special target **.POSIX**
  - omit any special target reserved for implementations (a leading period followed by upper-case letters) that has not been specified by this section.

18469       The behaviour of *make* is unspecified if either or both of these conditions are not met.18470 **OPTIONS**18471       The *make* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

18472       The following options are supported:

18473       **-e**       Cause environment variables, including those with null values, to override macro  
18474       assignments within makefiles.18475       **-f *makefile***18476       Specify a different makefile. The argument *makefile* is a pathname of a description file,  
18477       which is also referred to as the *makefile*. A pathname of "-" denotes the standard input.  
18478       There can be multiple instances of this option, and they will be processed in the order  
18479       specified. The effect of specifying the same option-argument more than once is  
18480       unspecified.18481       **-i**       Ignore error codes returned by invoked commands. This mode is the same as if the  
18482       special target **.IGNORE** were specified without prerequisites.18483       **-k**       Continue to update other targets that do not depend on the current target if a non-  
18484       ignored error occurs while executing the commands to bring a target up-to-date.18485       **-n**       Write commands that would be executed on standard output, but do not execute them.  
18486       However, lines with a plus sign (+) prefix will be executed. In this mode, lines with an  
18487       at sign (@) character prefix will be written to standard output.

18488       **-p**       Write to standard output the complete set of macro definitions and target descriptions.  
 18489               The output format is unspecified.

18490       **-q**       Return a zero exit value if the target file is up-to-date; otherwise, return an exit value of  
 18491               1. Targets will not be updated if this option is specified. However, a command line  
 18492               (associated with the targets) with a plus sign (+) prefix will be executed.

18493       **-r**       Clear the suffix list and do not use the built-in rules.

18494       **-S**       Terminate *make* if an error occurs while executing the commands to bring a target up-  
 18495               to-date. This will be the default and the opposite of **-k**.

18496       **-s**       Do not write command lines or touch messages (see **-t**) to standard output before  
 18497               executing. This mode is the same as if the special target **.SILENT** were specified  
 18498               without prerequisites.

18499       **-t**       Update the modification time of each target as though a *touch target* had been executed.  
 18500               Targets that have prerequisites but no commands (see **Target Rules** on page 507), or  
 18501               that are already up-to-date, will not be touched in this manner. Write messages to  
 18502               standard output for each target file indicating the name of the file and that it was  
 18503               touched. Normally, the command lines associated with each target are not executed.  
 18504               However, a command line with a plus sign (+) prefix will be executed.

18505       If the **-k** and **-S** options are both specified on the command line, by the **MAKEFLAGS**  
 18506       environment variable, or by the **MAKEFLAGS** macro, the last one evaluated will take  
 18507       precedence. The **MAKEFLAGS** environment variable will be evaluated first and the command  
 18508       line will be evaluated second. Assignments to the **MAKEFLAGS** macro will be evaluated as  
 18509       described in the ENVIRONMENT VARIABLES section.

## 18510 OPERANDS

18511       The following operands are supported:

18512       *target\_name*  
 18513               Target names, as defined in the EXTENDED DESCRIPTION section. If no target is  
 18514               specified, while *make* is processing the makefiles, the first target that *make* encounters  
 18515               that is not a special target or an inference rule will be used.

18516       *macro=name*  
 18517               Macro definitions, as defined in **Macros** on page 509.

18518       If the *target\_name* and *macro=name* operands are intermixed on the command line, the results are  
 18519       unspecified.

## 18520 STDIN

18521       The standard input will be used only if the *makefile* option-argument is "-". See the INPUT FILES  
 18522       section.

## 18523 INPUT FILES

18524       The input file, otherwise known as the makefile, is a text file containing rules, macro definitions  
 18525       and comments.

## 18526 ENVIRONMENT VARIABLES

18527       The following environment variables affect the execution of *make*:

18528       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 18529               **LANG** is unset or null, the corresponding value from the implementation-dependent  
 18530               default locale will be used. If any of the internationalisation variables contains an  
 18531               invalid setting, the utility will behave as if none of the variables had been defined.



|          |                    |                                                                                                            |
|----------|--------------------|------------------------------------------------------------------------------------------------------------|
| 18532    | <i>LC_ALL</i>      |                                                                                                            |
| 18533    |                    | If set to a non-empty string value, override the values of all the other                                   |
| 18534    |                    | internationalisation variables.                                                                            |
| 18535    | <i>LC_CTYPE</i>    |                                                                                                            |
| 18536    |                    | Determine the locale for the interpretation of sequences of bytes of text data as                          |
| 18537    |                    | characters (for example, single- as opposed to multi-byte characters in arguments and                      |
| 18538    |                    | input files).                                                                                              |
| 18539    | <i>LC_MESSAGES</i> |                                                                                                            |
| 18540    |                    | Determine the locale that should be used to affect the format and contents of diagnostic                   |
| 18541    |                    | messages written to standard error.                                                                        |
| 18542    | <i>MAKEFLAGS</i>   |                                                                                                            |
| 18543    |                    | This variable is interpreted as a character string representing a series of option                         |
| 18544    |                    | characters to be used as the default options. The implementation will accept both of                       |
| 18545    |                    | the following formats (but need not accept them when intermixed):                                          |
| 18546    |                    | 1. The characters are option letters without the leading hyphens or blank character                        |
| 18547    |                    | separation used on a command line.                                                                         |
| 18548    |                    | 2. The characters are formatted in a manner similar to a portion of the <i>make</i>                        |
| 18549    |                    | command line: options are preceded by hyphens and blank-character-separated                                |
| 18550    |                    | as described in the <b>XBD</b> specification, <b>Section 10.2, Utility Syntax Guidelines</b> .             |
| 18551    |                    | The <i>macro=name</i> macro definition operands can also be included. The difference                       |
| 18552    |                    | between the contents of <i>MAKEFLAGS</i> and the command line is that the contents                         |
| 18553    |                    | of the variable will not be subjected to the word expansions (see Section 2.6 on                           |
| 18554    |                    | page 31) associated with parsing the command line values.                                                  |
| 18555    |                    | When the command-line options <b>-f</b> or <b>-p</b> are used, they will take effect regardless of         |
| 18556    |                    | whether they also appear in <i>MAKEFLAGS</i> . If they otherwise appear in <i>MAKEFLAGS</i> ,              |
| 18557    |                    | the result is undefined.                                                                                   |
| 18558    |                    | The <i>MAKEFLAGS</i> variable will be accessed from the environment before the makefile                    |
| 18559    |                    | is read. At that time, all of the options (except <b>-f</b> and <b>-p</b> ) and command-line macros        |
| 18560    |                    | not already included in <i>MAKEFLAGS</i> are added to the <i>MAKEFLAGS</i> macro. The                      |
| 18561    |                    | <i>MAKEFLAGS</i> macro will be passed into the environment as an environment variable                      |
| 18562    |                    | for all child processes. If the <i>MAKEFLAGS</i> macro is subsequently set by the makefile,                |
| 18563    |                    | it replaces the <i>MAKEFLAGS</i> variable currently found in the environment.                              |
| 18564 EX | <i>NLSPATH</i>     |                                                                                                            |
| 18565    |                    | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                    |
| 18566 EX | <i>PROJECTDIR</i>  |                                                                                                            |
| 18567    |                    | Provide a directory to be used to search for SCCS files not found in the current                           |
| 18568    |                    | directory. In all of the following cases, the search for SCCS files will be made in the                    |
| 18569    |                    | directory <b>SCCS</b> in the identified directory. If the value of <i>PROJECTDIR</i> begins with a         |
| 18570    |                    | slash, it is considered an absolute pathname; otherwise, the home directory of a user of                   |
| 18571    |                    | that name is examined for a subdirectory <b>src</b> or <b>source</b> . If such a directory is found, it is |
| 18572    |                    | used. Otherwise, the value is used as a relative pathname.                                                 |
| 18573    |                    | If <i>PROJECTDIR</i> is not set or has a null value, the search for SCCS files will be made in             |
| 18574    |                    | the directory <b>SCCS</b> in the current directory.                                                        |
| 18575    |                    | The setting of <i>PROJECTDIR</i> affects all files listed in the remainder of this utility                 |
| 18576    |                    | description for files with a component named <b>SCCS</b> .                                                 |

18577 The value of the *SHELL* environment variable will not be used as a macro and will not be  
 18578 modified by defining the **SHELL** macro in a makefile or on the command line. All other  
 18579 environment variables, including those with null values, are used as macros, as defined in  
 18580 **Macros** on page 509.

#### 18581 ASYNCHRONOUS EVENTS

18582 If not already ignored, *make* will trap SIGHUP, SIGTERM, SIGINT and SIGQUIT and remove the  
 18583 current target unless the target is a directory or the target is a prerequisite of the special target  
 18584 **.PRECIOUS** or unless one of the **-n**, **-p** or **-q** options was specified. Any targets removed in  
 18585 this manner will be reported in diagnostic messages of unspecified format, written to standard  
 18586 error. After this cleanup process, if any, *make* will take the standard action for all other signals.

#### 18587 STDOUT

18588 The *make* utility will write all commands to be executed to standard output unless the **-s** option  
 18589 was specified, the command is prefixed with an at sign, or the special target **.SILENT** has either  
 18590 the current target as a prerequisite or has no prerequisites. If *make* is invoked without any work  
 18591 needing to be done, it will write a message to standard output indicating that no action was  
 18592 taken.

#### 18593 STDERR

18594 Used only for diagnostic messages.

#### 18595 OUTPUT FILES

18596 None. However, utilities invoked by *make* may create additional files.

#### 18597 EXTENDED DESCRIPTION

18598 The *make* utility attempts to perform the actions required to ensure that the specified targets are  
 18599 up-to-date. A target is considered out-of-date if it is older than any of its prerequisites or if it  
 18600 does not exist. The *make* utility treats all prerequisites as targets themselves and recursively  
 18601 ensures that they are up-to-date, processing them in the order in which they appear in the rule.  
 18602 The *make* utility uses the modification times of files to determine if the corresponding targets are  
 18603 out-of-date.

18604 After *make* has ensured that all of the prerequisites of a target are up-to-date and if the target is  
 18605 out-of-date, the commands associated with the target entry are executed. If there are no  
 18606 commands listed for the target, the target is treated as up-to-date.

#### 18607 Makefile Syntax

18608 A makefile can contain rules, macro definitions (see **Macros** on page 509), and comments. There  
 18609 are two kinds of rules: inference rules and target rules. The *make* utility contains a set of built-in  
 18610 inference rules. If the **-r** option is present, the built-in rules are not used and the suffix list is  
 18611 cleared. Additional rules of both types can be specified in a makefile. If a rule or macro is  
 18612 defined more than once, the value of the rule or macro will be that of the last one specified.  
 18613 Comments start with a number sign (#) and continue until an unescaped newline character is  
 18614 reached.

18615 EX By default, the following files are tried in sequence: **./makefile**, **./Makefile**, **./s.makefile**,  
 18616 **SCCS/s.makefile**, **./s.Makefile** and **SCCS/s.Makefile**.

18617 The **-f** option directs *make* to ignore any of these default files and use the specified argument as a  
 18618 makefile instead. If the **"-"** argument is specified, standard input will be used.

18619 The term *makefile* is used to refer to any rules provided by the user, whether in **./makefile** or its  
 18620 variants, or specified by the **-f** option.

18621 The rules in makefiles consist of the following types of lines: target rules, including special  
 18622 targets (see **Target Rules** on page 507); inference rules (see **Inference Rules** on page 510); macro

18623 definitions (see **Macros** on page 509); empty lines; and comments. Comments start with a  
 18624 number sign (#) and continue until an unescaped newline character is reached.

18625 When an escaped newline character (one preceded by a **backslash**) is found anywhere in the  
 18626 makefile, it is replaced, along with any leading white space on the following line, with a single  
 18627 space character.

## 18628 **Makefile Execution**

18629 Command lines are processed one at a time by writing the command line to the standard output  
 18630 (unless one of the conditions listed below under "@" suppresses the writing) and executing the  
 18631 commands in the line. A tab character may precede the command to standard output.  
 18632 Commands will be executed by passing the command line to the command interpreter in the  
 18633 same manner as if the string were the argument to the **XSH** specification *system()* function.

18634 The environment for the command being executed will contain all of the variables in the  
 18635 environment of *make*. The macros from the command line to *make* will be added to *make*'s  
 18636 environment. Other implementation-dependent variables may also be added to *make*'s  
 18637 environment. If any command-line macro has been defined elsewhere, the command-line value  
 18638 will overwrite the existing value. If the **MAKEFLAGS** variable is not set in the environment in  
 18639 which *make* was invoked, in the makefile or on the command line, it will be created by *make*, and  
 18640 will contain all options specified on the command line except for the **-f** and **-p** options. It may  
 18641 also contain implementation-dependent options.

18642 By default, when *make* receives a non-zero status from the execution of a command, it terminates  
 18643 with an error message to standard error.

18644 Command lines can have one or more of the following prefixes: a hyphen (-), an at sign (@), or a  
 18645 plus sign (+). These modify the way in which *make* processes the command. When a command  
 18646 is written to standard output, the prefix is not included in the output.

18647 - If the command prefix contains a hyphen, or the **-i** option is present, or the special target  
 18648 **.IGNORE** has either the current target as a prerequisite or has no prerequisites, any error  
 18649 found while executing the command will be ignored.

18650 @ If the command prefix contains an at sign and the command-line **-n** option is not specified,  
 18651 or the **-s** option is present, or the special target **.SILENT** has either the current target as a  
 18652 prerequisite or has no prerequisites, the command will not be written to standard output  
 18653 before it is executed.

18654 + If the command prefix contains a plus sign, this indicates a command line that will be  
 18655 executed even if **-n**, **-q** or **-t** is specified.

## 18656 **Target Rules**

18657 Target rules are formatted as follows:

```
18658 target [target...]: [prerequisite...][;command]
18659 [<tab>command
18660 <tab>command
18661 ...]
```

18662 *line that does not begin with <tab>*

18663 Target entries are specified by a blank-character-separated, non-null list of targets, then a colon,  
 18664 then a blank-character-separated, possibly empty list of prerequisites. Text following a  
 18665 semicolon, if any, and all following lines that begin with a tab character, are command lines to be  
 18666 executed to update the target. The first non-empty line that does not begin with a tab character

|          |           |                                                                                                           |  |
|----------|-----------|-----------------------------------------------------------------------------------------------------------|--|
| 18667    |           | or "#" begins a new entry. An empty or blank line, or a line beginning with "#", may begin a new          |  |
| 18668    |           | entry.                                                                                                    |  |
| 18669    |           | Applications must select target names from the set of characters consisting solely of periods,            |  |
| 18670    |           | underscores, digits and alphabets from the portable character set (see the <b>XBD</b> specification,      |  |
| 18671    |           | <b>Section 4.1, Portable Character Set</b> ). Implementations may allow other characters in target        |  |
| 18672    |           | names as extensions. The interpretation of targets containing the characters "%" and "\"" is              |  |
| 18673    |           | implementation-dependent.                                                                                 |  |
| 18674    |           | A target that has prerequisites, but does not have any commands, can be used to add to the                |  |
| 18675    |           | prerequisite list for that target. Only one target rule for any given target can contain commands.        |  |
| 18676    |           | Lines that begin with one of the following are called <i>special targets</i> and control the operation of |  |
| 18677    |           | <i>make</i> :                                                                                             |  |
| 18678    | .DEFAULT  | If the makefile uses this special target, it must be specified with commands,                             |  |
| 18679    |           | but without prerequisites. The commands will be used by <i>make</i> if there are no                       |  |
| 18680    |           | other rules available to build a target.                                                                  |  |
| 18681    | .IGNORE   | Prerequisites of this special target are targets themselves; this will cause errors                       |  |
| 18682    |           | from commands associated with them to be ignored in the same manner as                                    |  |
| 18683    |           | specified by the <b>-i</b> option. Subsequent occurrences of <b>.IGNORE</b> add to the                    |  |
| 18684    |           | list of targets ignoring command errors. If no prerequisites are specified, <i>make</i>                   |  |
| 18685    |           | will behave as if the <b>-i</b> option had been specified and errors from all                             |  |
| 18686    |           | commands associated with all targets will be ignored.                                                     |  |
| 18687    | .POSIX    | This special target must be specified without prerequisites or commands. If it                            |  |
| 18688    |           | appears before the first non-comment line in the makefile, <i>make</i> will process                       |  |
| 18689    |           | the makefile as specified by this section; otherwise, the behaviour of <i>make</i> is                     |  |
| 18690    |           | unspecified.                                                                                              |  |
| 18691    | .PRECIOUS | Prerequisites of this special target will not be removed if <i>make</i> receives one of                   |  |
| 18692    |           | the asynchronous events explicitly described in the <b>ASYNCHRONOUS</b>                                   |  |
| 18693    |           | <b>EVENTS</b> section. Subsequent occurrences of <b>.PRECIOUS</b> add to the list of                      |  |
| 18694    |           | precious files. If no prerequisites are specified, all targets in the makefile will                       |  |
| 18695    |           | be treated as if specified with <b>.PRECIOUS</b> .                                                        |  |
| 18696 EX | .SCCS_GET | This special target must be specified without prerequisites. If this special                              |  |
| 18697    |           | target is included in a makefile, the commands specified with this target                                 |  |
| 18698    |           | replace the default commands associated with this special target. (See                                    |  |
| 18699    |           | <b>Default Rules</b> on page 513.) The commands specified with this target are                            |  |
| 18700    |           | used to get all SCCS files that are not found in the current directory.                                   |  |
| 18701    |           | When source files are named in a dependency list, <i>make</i> treats them just like                       |  |
| 18702    |           | any other target. Because the source file is presumed to be present in the                                |  |
| 18703    |           | directory, there is no need to add an entry for it to the makefile. When a target                         |  |
| 18704    |           | has no dependencies, but is present in the directory, <i>make</i> assumes that that                       |  |
| 18705    |           | file is up-to-date. If, however, an SCCS file named <b>SCCS/s.source_file</b> is found                    |  |
| 18706    |           | for a target <i>source_file</i> , <i>make</i> does some additional checking to assure that the            |  |
| 18707    |           | target is up-to-date. If the target is missing, or if the SCCS file is newer, <i>make</i>                 |  |
| 18708    |           | automatically issues the commands specified for the <b>.SCCS_GET</b> special                              |  |
| 18709    |           | target to retrieve the most recent version. However, if the target is writable by                         |  |
| 18710    |           | anyone, <i>make</i> does not retrieve a new version.                                                      |  |
| 18711    | .SILENT   | Prerequisites of this special target are targets themselves; this causes                                  |  |
| 18712    |           | commands associated with them to not be written to the standard output                                    |  |
| 18713    |           | before they are executed. Subsequent occurrences of <b>.SILENT</b> add to the list                        |  |

18714 of targets with silent commands. If no prerequisites are specified, *make* will  
 18715 behave as if the `-s` option had been specified and no commands or touch  
 18716 messages associated with any target will be written to standard output.

18717 **.SUFFIXES** Prerequisites of **.SUFFIXES** are appended to the list of known suffixes and are  
 18718 used in conjunction with the inference rules (see **Inference Rules** on page  
 18719 510). If **.SUFFIXES** does not have any prerequisites, the list of known suffixes  
 18720 will be cleared. Makefiles must not associate commands with **.SUFFIXES**.

18721 Targets with names consisting of a leading period followed by the upper-case letters **POSIX** and  
 18722 then any other characters are reserved for future standardisation. Targets with names consisting  
 18723 of a leading period followed by one or more upper-case letters are reserved for implementation  
 18724 extensions.

18725 **Macros**

18726 Macro definitions are in the form:

18727 `string1 "=" [string2]`

18728 The macro named *string1* is defined as having the value of *string2*, where *string2* is defined as all  
 18729 characters, if any, after the equal sign, up to a comment character (#) or an unescaped newline  
 18730 character. Any blank characters immediately before or after the equal sign will be ignored.

18731 Subsequent appearances of `$(string1)` or `${string1}` are replaced by *string2*. The parentheses or  
 18732 braces are optional if *string1* is a single character. The macro `$$` is replaced by the single  
 18733 character "\$".

18734 Applications must select macro names from the set of characters consisting solely of periods,  
 18735 underscores, digits and alphabets from the portable character set (see the **XBD** specification,  
 18736 **Section 4.1, Portable Character Set**). A macro name cannot contain an equal sign.  
 18737 Implementations may allow other characters in macro names as extensions.

18738 Macros can appear anywhere in the makefile. Macros in target lines will be evaluated when the  
 18739 target line is read. Macros in command lines will be evaluated when the command is executed.  
 18740 Macros in macro definition lines will not be evaluated until the new macro being defined is used  
 18741 in a rule or command. A macro that has not been defined will evaluate to a null string without  
 18742 causing any error condition.

18743 The forms `$(string1[:subst1=[subst2]])` or `${string1[:subst1=[subst2]]}` can be used to replace all  
 18744 occurrences of *subst1* with *subst2* when the macro substitution is performed. The *subst1* to be  
 18745 replaced is recognised when it is a suffix at the end of a word in *string1* (where a *word*, in this  
 18746 context, is defined to be a string delimited by the beginning of the line, a blank or newline  
 18747 character).

18748 Macro assignments will be accepted from the sources listed below, in the order shown. If a  
 18749 macro name already exists at the time it is being processed, the newer definition will replace the  
 18750 existing definition.

- 18751 1. Macros defined in *make*'s built-in inference rules.
- 18752 2. The contents of the environment, including the variables with null values, in the order  
 18753 defined in the environment.
- 18754 3. Macros defined in the makefiles, processed in the order specified.
- 18755 4. Macros specified on the command line. It is unspecified whether the internal macros  
 18756 defined in **Internal Macros** on page 511 are accepted from the command line.

18757 If the `-e` option is specified, the order of processing sources items 2 and 3 will be reversed.

18758 The **SHELL** macro is treated specially. It is provided by *make* and set to the pathname of the  
 18759 shell command language interpreter (see *sh*). The *SHELL* environment variable will not affect  
 18760 the value of the **SHELL** macro. If **SHELL** is defined in the makefile or is specified on the  
 18761 command line, it will replace the original value of the **SHELL** macro, but will not affect the  
 18762 *SHELL* environment variable. Other effects of defining **SHELL** in the makefile or on the  
 18763 command line are implementation-dependent.

## 18764 Inference Rules

18765 Inference rules are formatted as follows:

```
18766 target:
18767 <tab>command
18768 [<tab>command]
18769 ...
18770 line that does not begin with <tab> or #
```

18771 The *target* portion must be a valid target name (see **Target Rules** on page 507) of the form *.s2* or  
 18772 *.s1.s2* (where *.s1* and *.s2* are suffixes that have been given as prerequisites of the **.SUFFIXES**  
 18773 special target and *s1* and *s2* do not contain any slashes or periods.) If there is only one period in  
 18774 the target, it is a single-suffix inference rule. Targets with two periods are double-suffix  
 18775 inference rules. Inference rules can have only one target before the colon.

18776 The makefile must not specify prerequisites for inference rules; no characters other than white  
 18777 space can follow the colon in the first line, except when creating the *empty rule*, described below.  
 18778 Prerequisites are inferred, as described below.

18779 Inference rules can be redefined. A target that matches an existing inference rule will overwrite  
 18780 the old inference rule. An empty rule can be created with a command consisting of simply a  
 18781 semicolon (that is, the rule still exists and is found during inference rule search, but since it is  
 18782 empty, execution has no effect). The empty rule also can be formatted as follows:

```
18783 rule: ;
```

18784 where zero or more blank characters separate the colon and semicolon.

18785 The *make* utility uses the suffixes of targets and their prerequisites to infer how a target can be  
 18786 made up-to-date. A list of inference rules defines the commands to be executed. By default,  
 18787 *make* contains a built-in set of inference rules. Additional rules can be specified in the makefile.

18788 The special target **.SUFFIXES** contains as its prerequisites a list of suffixes that are to be used by  
 18789 the inference rules. The order in which the suffixes are specified defines the order in which the  
 18790 inference rules for the suffixes are used. New suffixes will be appended to the current list by  
 18791 specifying a **.SUFFIXES** special target in the makefile. A **.SUFFIXES** target with no prerequisites  
 18792 will clear the list of suffixes. An empty **.SUFFIXES** target followed by a new **.SUFFIXES** list is  
 18793 required to change the order of the suffixes.

18794 Normally, the user would provide an inference rule for each suffix. The inference rule to update  
 18795 a target with a suffix *.s1* from a prerequisite with a suffix *.s2* is specified as a target *.s2.s1*. The  
 18796 internal macros provide the means to specify general inference rules. (See **Internal Macros** on  
 18797 page 511.)

18798 When no target rule is found to update a target, the inference rules are checked. The suffix of the  
 18799 target (*.s1*) to be built is compared to the list of suffixes specified by the **.SUFFIXES** special  
 18800 targets. If the *.s1* suffix is found in **.SUFFIXES**, the inference rules are searched in the order  
 18801 defined for the first *.s2.s1* rule whose prerequisite file (*\$\*.s2*) exists. If the target is out-of-date

18802 with respect to this prerequisite, the commands for that inference rule are executed.

18803 If the target to be built does not contain a suffix and there is no rule for the target, the single  
 18804 suffix inference rules will be checked. The single-suffix inference rules define how to build a  
 18805 target if a file is found with a name that matches the target name with one of the single suffixes  
 18806 appended. A rule with one suffix *.s2* is the definition of how to build *target* from *target.s2*. The  
 18807 other suffix (*.s1*) is treated as null.

18808 EX A tilde (~) in the above rules refers to an SCCS file in the current directory. Thus, the rule *.c~.o*  
 18809 would transform an SCCS C-language source file into an object file (*.o*). Because the *s.* of the  
 18810 SCCS files is a prefix, it is incompatible with *make's* suffix point of view. Hence, the ~ is a way of  
 18811 changing any file reference into an SCCS file reference.

## 18812 Libraries

18813 If a target or prerequisite contains parentheses, it will be treated as a member of an archive  
 18814 library. For the *lib(member.o)* expression *lib* refers to the name of the archive library and *member.o*  
 18815 to the member name. The member must be an object file with the *.o* suffix. The modification  
 18816 time of the expression is the modification time for the member as kept in the archive library. See  
 18817 *ar*. The *.a* suffix refers to an archive library. The *.s2.a* rule is used to update a member in the  
 18818 library from a file with a suffix *.s2*.

## 18819 Internal Macros

18820 The *make* utility maintains five internal macros that can be used in target and inference rules. In  
 18821 order to clearly define the meaning of these macros, some clarification of the terms *target rule*,  
 18822 *inference rule*, *target* and *prerequisite* is necessary.

18823 Target rules are specified by the user in a makefile for a particular target. Inference rules are  
 18824 user- or *make*-specified rules for a particular class of target names. Explicit prerequisites are  
 18825 those prerequisites specified in a makefile on target lines. Implicit prerequisites are those  
 18826 prerequisites that are generated when inference rules are used. Inference rules are applied to  
 18827 implicit prerequisites or to explicit prerequisites that do not have target rules defined for them in  
 18828 the makefile. Target rules are applied to targets specified in the makefile.

18829 Before any target in the makefile is updated, each of its prerequisites (both explicit and implicit)  
 18830 will be updated. This is accomplished by recursively processing each prerequisite. Upon  
 18831 recursion, each prerequisite becomes a target itself. Its prerequisites in turn are processed  
 18832 recursively until a target is found that has no prerequisites, at which point the recursion stops.  
 18833 The recursion then backs up, updating each target as it goes.

18834 In the definitions that follow, the word *target* refers to one of:

- 18835 • a target specified in the makefile
- 18836 • an explicit prerequisite specified in the makefile that becomes the target when *make* processes  
 18837 it during recursion
- 18838 • an implicit prerequisite that becomes a target when *make* processes it during recursion.

18839 In the definitions that follow, the word *prerequisite* refers to one of the following:

- 18840 • an explicit prerequisite specified in the makefile for a particular target
- 18841 • an implicit prerequisite generated as a result of locating an appropriate inference rule and  
 18842 corresponding file that matches the suffix of the target.

18843           The five internal macros are:

18844       \$@       The \$@ evaluates to the full target name of the current target, or the archive filename  
18845               part of a library archive target. It is evaluated for both target and inference rules.

18846               For example, in the **.c.a** inference rule, \$@ represents the out-of-date **.a** file to be built.  
18847               Similarly, in a makefile target rule to build **lib.a** from **file.c**, \$@ represents the out-of-  
18848               date **lib.a**.

18849       \$%       The \$% macro is evaluated only when the current target is an archive library member  
18850               of the form *libname(member.o)*. In these cases, \$@ evaluates to *libname* and \$% evaluates  
18851               to *member.o*. The \$% macro is evaluated for both target and inference rules.

18852               For example, in a makefile target rule to build **lib.a(file.o)**, \$% represents **file.o**  
18853               as opposed to \$@, which represents **lib.a**.

18854       \$?       The \$? macro evaluates to the list of prerequisites that are newer than the current  
18855               target. It is evaluated for both target and inference rules.

18856               For example, in a makefile target rule to build **prog** from **file1.o**, **file2.o** and **file3.o**, and  
18857               where **prog** is not out of date with respect to **file1.o**, but is out of date with respect to  
18858               **file2.o** and **file3.o**, \$? represents **file2.o** and **file3.o**.

18859       \$<       In an inference rule, \$< evaluates to the filename whose existence allowed the inference  
18860               rule to be chosen for the target. In the **.DEFAULT** rule, the \$< macro evaluates to the  
18861               current target name. The \$< macro is evaluated only for inference rules.

18862               For example, in the **.c.a** inference rule, \$< represents the prerequisite **.c** file.

18863       \$\*       The \$\* macro evaluates to the current target name with its suffix deleted. It is  
18864               evaluated at least for inference rules.

18865               For example, in the **.c.a** inference rule, \$\*.o represents the out-of-date **.o** file that  
18866               corresponds to the prerequisite **.c** file.

18867       Each of the internal macros has an alternative form. When an upper-case D or F is appended to  
18868       any of the macros, the meaning is changed to the *directory part* for D and *filename part* for F. The  
18869       directory part is the path prefix of the file without a trailing slash; for the current directory, the  
18870       directory part is “.”. When the \$? macro contains more than one prerequisite filename, the \$(?D)  
18871       and \$(?F) (or \${?D} and \${?F}) macros expand to a list of directory name parts and filename parts  
18872       respectively.

18873       For the target *lib(member.o)* and the **s2.a** rule, the internal macros are defined as:

18874       \$<       *member.s2*

18875       \$\*       *member*

18876       \$@       *lib*

18877       \$?       *member.s2*

18878       \$%       *member.o*



18879 **Default Rules**

18880 The default rules for *make* achieve results that are the same as if the following were used.  
 18881 Implementations that do not support FORTRAN may omit **FC**, **FFLAGS** and the **.f** inference  
 18882 rules. Implementations may provide additional macros and rules.

18883 **SPECIAL TARGETS**

18884 EX **.SCCS\_GET:** sccs \$(SCCSFLAGS) get \$(SCCSGETFLAGS) \$@

18885 EX **.SUFFIXES:** .o .c .y l .a .sh .f .c~ .y~ .l~ .sh~ .f~

18886 **MACROS**

18887 MAKE=make

18888 AR=ar

18889 ARFLAGS=-rv

18890 YACC=yacc

18891 YFLAGS=

18892 LEX=lex

18893 LFLAGS=

18894 LDFLAGS=

18895 CC=c89

18896 CFLAGS=-O

18897 FC=fort77

18898 FFLAGS=-O 1

18899 EX GET=get

18900 GFLAGS=

18901 SCCSFLAGS=

18902 SCCSGETFLAGS=-s

18903 **SINGLE SUFFIX RULES**

18904 **.c:**

18905 \$(CC) \$(CFLAGS) \$(LDFLAGS) -o \$@ \$<

18906 **.f:**

18907 \$(FC) \$(FFLAGS) \$(LDFLAGS) -o \$@ \$<

18908 **.sh:**

18909 cp \$< \$@

18910 chmod a+x \$@

18911 EX **.c~:**

18912 \$(GET) \$(GFLAGS) -p \$< > \$\*.c

18913 \$(CC) \$(CFLAGS) \$(LDFLAGS) -o \$@ \$\*.c

18914 **.f~:**

18915 \$(GET) \$(GFLAGS) -p \$< > \$\*.f

18916 \$(FC) \$(FFLAGS) \$(LDFLAGS) -o \$@ \$\*.f

18917 **.sh~:**

18918 \$(GET) \$(GFLAGS) -p \$< > \$\*.sh

18919 cp \$\*.sh \$@

18920 chmod a+x \$@

18921 **DOUBLE SUFFIX RULES**

18922 **.c.o:**

18923 \$(CC) \$(CFLAGS) -c \$<

18924 **.f.o:**

18925 \$(FC) \$(FFLAGS) -c \$<

18926 **.y.o:**

18927 \$(YACC) \$(YFLAGS) \$<

18928 \$(CC) \$(CFLAGS) -c y.tab.c

18929 rm -f y.tab.c

```

18930 mv y.tab.o $@
18931 .l.o:
18932 $(LEX) $(LFLAGS) $<
18933 $(CC) $(CFLAGS) -c lex.yy.c
18934 rm -f lex.yy.c
18935 mv lex.yy.o $@
18936 .y.c:
18937 $(YACC) $(YFLAGS) $<
18938 mv y.tab.c $@
18939 .l.c:
18940 $(LEX) $(LFLAGS) $<
18941 mv lex.yy.c $@
18942 EX .c~.o:
18943 $(GET) $(GFLAGS) -p $< > $*.c
18944 $(CC) $(CFLAGS) -c $*.c
18945 .f~.o:
18946 $(GET) $(GFLAGS) -p $< > $*.f
18947 $(FC) $(FFLAGS) -c $*.f
18948 .y~.o:
18949 $(GET) $(GFLAGS) -p $< > $*.y
18950 $(YACC) $(YFLAGS) $*.y
18951 $(CC) $(CFLAGS) -c y.tab.c
18952 rm -f y.tab.c
18953 mv y.tab.o $@
18954 .l~.o:
18955 $(GET) $(GFLAGS) -p $< > $*.l
18956 $(LEX) $(LFLAGS) $*.l
18957 $(CC) $(CFLAGS) -c lex.yy.c
18958 rm -f lex.yy.c
18959 mv lex.yy.o $@
18960 .y~.c:
18961 $(GET) $(GFLAGS) -p $< > $*.y
18962 $(YACC) $(YFLAGS) $*.y
18963 mv y.tab.c $@
18964 .l~.c:
18965 $(GET) $(GFLAGS) -p $< > $*.l
18966 $(LEX) $(LFLAGS) $*.l
18967 mv lex.yy.c $@
18968 .c.a:
18969 $(CC) -c $(CFLAGS) $<
18970 $(AR) $(ARFLAGS) $@ $*.o
18971 rm -f $*.o
18972 .f.a:
18973 $(FC) -c $(FFLAGS) $<
18974 $(AR) $(ARFLAGS) $@ $*.o
18975 rm -f $*.o

```

#### 18976 EXIT STATUS

18977 When the **-q** option is specified, the *make* utility will exit with one of the following values:

- 18978 0 Successful completion.
- 18979 1 The target was not up-to-date.
- 18980 >1 An error occurred.

18981 When the **-q** option is not specified, the *make* utility will exit with one of the following values:

18982 0 successful completion  
18983 >0 an error occurred

#### 18984 CONSEQUENCES OF ERRORS

18985 Default.

#### 18986 APPLICATION USAGE

18987 If there is a source file (such as *./source.c*) and there are two SCCS files corresponding to it  
18988 (*./s.source.c* and *./SCCS/s.source.c*), *make* will use the SCCS file in the current directory.  
18989 However, users are advised to use the underlying SCCS utilities (*admin*, *delta*, *get*, and so on) or  
18990 the *sccs* utility for all source files in a given directory. If both forms are used for a given source  
18991 file, future developers are very likely to be confused.

18992 It is incumbent upon portable makefiles to specify the **.POSIX** special target in order to  
18993 guarantee that they are not affected by local extensions.

18994 The **-k** and **-S** options are both present so that the relationship between the command line, the  
18995 *MAKEFLAGS* variable, and the makefile can be controlled precisely. If the **k** flag is passed in  
18996 *MAKEFLAGS* and a command is of the form:

18997 `$(MAKE) -S foo`

18998 then the default behaviour is restored for the child *make*.

18999 When the **-n** option is specified, it is always added to *MAKEFLAGS*. This allows a recursive  
19000 *make -n target* to be used to see all of the action that would be taken to update *target*.

19001 Because of widespread historical practice, interpreting a **#** number sign inside a variable as the  
19002 start of a comment has the unfortunate side effect of making it impossible to place a number  
19003 sign in a variable, thus forbidding something like:

19004 `CFLAGS = "-D COMMENT_CHAR=' #' "`

19005 Many historical *make* utilities stop chaining together inference rules when an intermediate target  
19006 is non-existent. For example, it might be possible for a *make* to determine that both *.y.c* and *.c.o*  
19007 could be used to convert a *.y* to a *.o*. Instead, in this case, *make* requires the use of a *.y.o* rule.

19008 The best way to provide portable makefiles is to include all of the rules needed in the makefile  
19009 itself. The rules provided use only features provided by other parts of the standard. The default  
19010 rules include rules for optional commands in the standard. Only rules pertaining to commands  
19011 that are provided are needed in an implementation's default set.

19012 Macros used within other macros are evaluated when the new macro is used rather than when  
19013 the new macro is defined. Therefore:

19014 `MACRO = value1`  
19015 `NEW = $(MACRO)`  
19016 `MACRO = value2`  
19017 `target:`  
19018 `echo $(NEW)`

19019 would produce *value2* and not *value1* since **NEW** was not expanded until it was needed in the  
19020 *echo* command line.

19021 Some historical applications have been known to intermix *target\_name* and *macro=name* operands  
19022 on the command line, expecting that all of the macros will be processed before any of the targets  
19023 are dealt with. Portable applications do not do this, although some backward compatibility  
19024 support may be included in some implementations.

19025 The following characters in filenames may give trouble:

19026 = : \ ' @

19027 For inference rules, the description of \$< and \$? seem similar. However, an example shows the  
19028 minor difference. In a makefile containing:

19029 foo.o: foo.h

19030 if **foo.h** is newer than **foo.o**, yet **foo.c** is older than **foo.o**, the built-in rule to make **foo.o** from  
19031 **foo.c** will be used, with \$< equal to **foo.c** and \$? equal to **foo.h**. If **foo.c** is also newer than **foo.o**,  
19032 \$< is equal to **foo.c** and \$? is equal to **foo.h foo.c**.

### 19033 EXAMPLES

19034 1. The following command:

19035 make

19036 makes the first target found in the makefile.

19037 2. The following command:

19038 make junk

19039 makes the target **junk**.

19040 3. The following makefile says that **pgm** depends on two files, **a.o** and **b.o**, and that they in  
19041 turn depend on their corresponding source files (**a.c** and **b.c**), and a common file **incl.h**:

```
19042 pgm: a.o b.o
19043 c89 a.o b.o -o pgm
19044 a.o: incl.h a.c
19045 c89 -c a.c
19046 b.o: incl.h b.c
19047 c89 -c b.c
```

19048 4. An example for making optimised **.o** files from **.c** files is:

```
19049 .c.o:
19050 c89 -c -O $*.c
```

19051 or:

```
19052 .c.o:
19053 c89 -c -O $<
```

19054 5. The most common use of the archive interface follows. Here, it is assumed that the source  
19055 files are all C-language source:

```
19056 lib: lib(file1.o) lib(file2.o) lib(file3.o)
19057 @echo lib is now up-to-date
```

19058 The **.c.a** rule is used to make **file1.o**, **file2.o** and **file3.o** and insert them into **lib**.

19059 The treatment of escaped newline characters throughout the makefile is historical practice.  
 19060 For example, the inference rule:

```
19061 .c.o\
19062 :
```

19063 works, and the macro:

```
19064 f= bar baz\
19065 biz
19066 a:
19067 echo ==$f==
```

19068 will echo ==bar baz biz==.

19069 If \$? were:

```
19070 /usr/include/stdio.h /usr/include/unistd.h foo.h
```

19071 then \$(?D) would be:

```
19072 /usr/include /usr/include .
```

19073 and \$(?F) would be:

```
19074 stdio.h unistd.h foo.h
```

19075 6. The contents of the built-in rules can be viewed by running:

```
19076 make -p -f /dev/null 2>/dev/null
```

#### 19077 FUTURE DIRECTIONS

19078 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 19079 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 19080 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 19081 finalised.

#### 19082 SEE ALSO

19083 *ar, c89, cc, get, lex, sh, yacc.*

#### 19084 CHANGE HISTORY

19085 First released in Issue 2.

#### 19086 Issue 4

19087 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 19088 Issue 4, Version 2

19089 Under **Default Rules**, the string `-G$@` is deleted from the line referencing `sccs`.

#### 19090 Issue 5

19091 FUTURE DIRECTIONS section added.

19092 **NAME**

19093       man — display system documentation

19094 **SYNOPSIS**19095       man [-k] *name*...19096 **DESCRIPTION**

19097       The *man* utility writes information about each of the *name* operands. If *name* is the name of a  
 19098       standard utility, *man* will at a minimum write a message describing the syntax used by the  
 19099       standard utility, its options and operands. If more information is available, the *man* utility will  
 19100       provide it in an implementation-dependent manner.

19101       An implementation may provide information for values of *name* other than the standard utilities.  
 19102       Standard utilities that are listed as optional and that are not supported by the implementation  
 19103       either will cause a brief message indicating that fact to be displayed or will cause a full display  
 19104       of information as described previously.

19105 **OPTIONS**19106       The *man* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

19107       The following option is supported:

19108       **-k**       Interpret *name* operands as keywords to be used in searching a utilities summary  
 19109       database that contains a brief purpose entry for each standard utility and write lines  
 19110       from the summary database that match any of the keywords. The keyword search  
 19111       produces results that are the equivalent of the output of the following command:

```
19112 grep -Ei '
19113 name
19114 name
19115 ...
19116 ' summary-database
```

19117       This assumes that the *summary-database* is a text file with a single entry per line; this  
 19118       organisation is not required and the example using *grep -Ei* is merely illustrative of the  
 19119       type of search intended. The purpose entry to be included in the database consists of a  
 19120       terse description of the purpose of the utility.

19121 **OPERANDS**

19122       The following operand is supported:

19123       *name*       A keyword or the name of a standard utility. When **-k** is not specified and *name* does  
 19124       not represent one of the standard utilities, the results are unspecified.

19125 **STDIN**

19126       Not used.

19127 **INPUT FILES**

19128       None.

19129 **ENVIRONMENT VARIABLES**19130       The following environment variables affect the execution of *man*:

19131       **LANG**       Provide a default value for the internationalisation variables that are unset or null. If  
 19132       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 19133       default locale will be used. If any of the internationalisation variables contains an  
 19134       invalid setting, the utility will behave as if none of the variables had been defined.

19135 **LC\_ALL**  
 19136 If set to a non-empty string value, override the values of all the other  
 19137 internationalisation variables.

19138 **LC\_CTYPE**  
 19139 Determine the locale for the interpretation of sequences of bytes of text data as  
 19140 characters (for example, single- as opposed to multi-byte characters in arguments and  
 19141 in the summary database). The value of *LC\_CTYPE* need not affect the format of the  
 19142 information written about the name operands.

19143 **LC\_MESSAGES**  
 19144 Determine the locale that should be used to affect the format and contents of diagnostic  
 19145 messages written to standard error and informative messages written to standard  
 19146 output.

19147 EX **NLSPATH**  
 19148 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

19149 **PAGER**  
 19150 Determine an output filtering command for writing the output to a terminal. Any  
 19151 string acceptable as a *command\_string* operand to the *sh -c* command is valid. When  
 19152 standard output is a terminal device, the manual-page output will be piped through the  
 19153 command. If the *PAGER* variable is null or not set, the command will be either *more* or  
 19154 another paginator utility documented in the system documentation.

19155 **ASYNCHRONOUS EVENTS**  
 19156 Default.

19157 **STDOUT**  
 19158 The *man* utility writes text describing the syntax of the utility *name*, its options and its operands,  
 19159 or, when *-k* is specified, lines from the summary database. The format of this text is  
 19160 implementation-dependent.

19161 **STDERR**  
 19162 Used only for diagnostic messages.

19163 **OUTPUT FILES**  
 19164 None.

19165 **EXTENDED DESCRIPTION**  
 19166 None.

19167 **EXIT STATUS**  
 19168 The following exit values are returned:  
 19169 0 Successful completion.  
 19170 >0 An error occurred.

19171 **CONSEQUENCES OF ERRORS**  
 19172 Default.

19173 **APPLICATION USAGE**  
 19174 None.

19175 **EXAMPLES**  
 19176 None.

19177 **FUTURE DIRECTIONS**  
 19178 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 19179 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the

|       |                                                                                             |  |
|-------|---------------------------------------------------------------------------------------------|--|
| 19180 | corrections. A future revision of this specification will align with IEEE Std. 1003.2b when |  |
| 19181 | finalised.                                                                                  |  |
| 19182 | <b>SEE ALSO</b>                                                                             |  |
| 19183 | <i>more.</i>                                                                                |  |
| 19184 | <b>CHANGE HISTORY</b>                                                                       |  |
| 19185 | First released in Issue 4.                                                                  |  |
| 19186 | <b>Issue 5</b>                                                                              |  |
| 19187 | FUTURE DIRECTIONS section added.                                                            |  |



19188 **NAME**

19189       mesg — permit or deny messages

19190 **SYNOPSIS**19191       mesg [*y*|*n*]19192 **DESCRIPTION**

19193       The *mesg* utility will control whether other users are allowed to send messages via *write*, *talk* or  
 19194       other utilities to a terminal device. The terminal device affected is determined by searching for  
 19195       the first terminal in the sequence of devices associated with standard input, standard output and  
 19196       standard error, respectively. With no arguments, *mesg* reports the current state without  
 19197       changing it. Processes with appropriate privileges may be able to send messages to the terminal  
 19198       independent of the current state.

19199 **OPTIONS**

19200       None.

19201 **OPERANDS**

19202       The following operands are supported in the POSIX locale:

19203       **y**       Grant permission to other users to send messages to the terminal device.19204       **n**       Deny permission to other users to send messages to the terminal device.19205 **STDIN**

19206       Not used.

19207 **INPUT FILES**

19208       None.

19209 **ENVIRONMENT VARIABLES**19210       The following environment variables affect the execution of *mesg*:

19211       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 19212       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 19213       default locale will be used. If any of the internationalisation variables contains an  
 19214       invalid setting, the utility will behave as if none of the variables had been defined.

19215       **LC\_ALL**

19216       If set to a non-empty string value, override the values of all the other  
 19217       internationalisation variables.

19218       **LC\_CTYPE**

19219       Determine the locale for the interpretation of sequences of bytes of text data as  
 19220       characters (for example, single- as opposed to multi-byte characters in arguments).

19221       **LC\_MESSAGES**

19222       Determine the locale that should be used to affect the format and contents of diagnostic  
 19223       messages written (by *mesg*) to standard error.

19224 EX       **NLSPATH**19225       Determine the location of message catalogues for the processing of **LC\_MESSAGES**.19226 **ASYNCHRONOUS EVENTS**

19227       Default.

19228 **STDOUT**19229       If no operand is specified, *mesg* displays the current terminal state in an unspecified format.

19230 **STDERR**

19231           Used only for diagnostic messages.

19232 **OUTPUT FILES**

19233           None.

19234 **EXTENDED DESCRIPTION**

19235           None.

19236 **EXIT STATUS**

19237           The following exit values are returned:

19238           0   Receiving messages is allowed.

19239           1   Receiving messages is not allowed.

19240           &gt;1  An error occurred.

19241 **CONSEQUENCES OF ERRORS**

19242           Default.

19243 **APPLICATION USAGE**

19244           The mechanism by which the message status of the terminal is changed is unspecified.

19245           Therefore, unspecified actions may cause the status of the terminal to change after *mesg* has successfully completed. These actions may include, but are not limited to: another invocation of19246           the *mesg* utility; login procedures; invocation of the *stty* utility; invocation of the *chmod* utility or *chmod()* function; and so on.19249 **EXAMPLES**

19250           None.

19251 **FUTURE DIRECTIONS**

19252           None.

19253 **SEE ALSO**19254           *talk*, *write*.19255 **CHANGE HISTORY**

19256           First released in Issue 2.

19257 **Issue 4**

19258           Aligned with the ISO/IEC 9945-2: 1993 standard.

19259 **NAME**

19260       mkdir — make directories

19261 **SYNOPSIS**19262       mkdir [-p][-m *mode*] *dir*...19263 **DESCRIPTION**19264       The *mkdir* utility will create the directories specified by the operands, in the order specified.19265       For each *dir* operand, the *mkdir* utility will perform actions equivalent to the **XSH** specification  
19266       *mkdir()* function, called with the following arguments:

- 19267       1. The *dir* operand is used as the *path* argument.
- 19268       2. The value of the bitwise inclusive OR of S\_IRWXU, S\_IRWXG and S\_IRWXO is used as the  
19269       *mode* argument. (If the **-m** option is specified, the *mode* option-argument overrides this  
19270       default.)

19271 **OPTIONS**19272       The *mkdir* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

19273       The following options are supported:

19274       **-m** *mode*

19275       Set the file permission bits of the newly-created directory to the specified *mode* value.  
 19276       The *mode* option-argument will be the same as the *mode* operand defined for the *chmod*  
 19277       utility. In the *symbolic\_mode* strings, the *op* characters "+" and "-" will be interpreted  
 19278       relative to an assumed initial mode of a=rwx; "+" will add permissions to the default  
 19279       mode, "-" will delete permissions from the default mode.

19280       **-p**       Create any missing intermediate pathname components.

19281       For each *dir* operand that does not name an existing directory, effects equivalent to  
 19282       those caused by the following command will occur:

```
19283 mkdir -p -m $(umask -S),u+wx $(dirname dir) &&
19284 mkdir [-m mode] dir
```

19285       where the **[-m *mode*]** option represents that option supplied to the original  
 19286       invocation of *mkdir*, if any.

19287       Each *dir* operand that names an existing directory will be ignored without error.19288 **OPERANDS**

19289       The following operand is supported:

19290       *dir*       A pathname of a directory to be created.19291 **STDIN**

19292       Not used.

19293 **INPUT FILES**

19294       None.

19295 **ENVIRONMENT VARIABLES**19296       The following environment variables affect the execution of *mkdir*:

19297       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 19298       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 19299       default locale will be used. If any of the internationalisation variables contains an  
 19300       invalid setting, the utility will behave as if none of the variables had been defined.

19301 *LC\_ALL*  
 19302 If set to a non-empty string value, override the values of all the other  
 19303 internationalisation variables.

19304 *LC\_CTYPE*  
 19305 Determine the locale for the interpretation of sequences of bytes of text data as  
 19306 characters (for example, single- as opposed to multi-byte characters in arguments).

19307 *LC\_MESSAGES*  
 19308 Determine the locale that should be used to affect the format and contents of diagnostic  
 19309 messages written to standard error.

19310 EX *NLSPATH*  
 19311 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

19312 **ASYNCHRONOUS EVENTS**  
 19313 Default.

19314 **STDOUT**  
 19315 Not used.

19316 **STDERR**  
 19317 Used only for diagnostic messages.

19318 **OUTPUT FILES**  
 19319 None.

19320 **EXTENDED DESCRIPTION**  
 19321 None.

19322 **EXIT STATUS**  
 19323 The following exit values are returned:  
 19324 0 All the specified directories were created successfully or the **-p** option was specified and all  
 19325 the specified directories now exist.  
 19326 >0 An error occurred.

19327 **CONSEQUENCES OF ERRORS**  
 19328 Default.

19329 **APPLICATION USAGE**  
 19330 The default file mode for directories is a=rwx (777 on most systems) with selected permissions  
 19331 removed in accordance with the file mode creation mask. For intermediate pathname  
 19332 components created by *mkdir*, the mode is the default modified by u+wx so that the  
 19333 subdirectories can always be created regardless of the file mode creation mask; if different  
 19334 ultimate permissions are desired for the intermediate directories, they can be changed  
 19335 afterwards with *chmod*.

19336 Note that some of the requested directories may have been created even if an error occurs.

19337 **EXAMPLES**  
 19338 None.

19339 **FUTURE DIRECTIONS**  
 19340 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 19341 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 19342 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 19343 finalised.

19344 **SEE ALSO**

19345           *rm*, *rmdir*, *umask*, the **XSH** specification description of *mkdir*().

19346 **CHANGE HISTORY**

19347           First released in Issue 2.

19348 **Issue 4**

19349           Aligned with the ISO/IEC 9945-2: 1993 standard.

19350 **Issue 5**

19351           FUTURE DIRECTIONS section added.

19352 **NAME**

19353 mkfifo — make FIFO special files

19354 **SYNOPSIS**19355 mkfifo [-m *mode*] *file*...19356 **DESCRIPTION**19357 The *mkfifo* utility will create the FIFO special files specified by the operands, in the order  
19358 specified.19359 For each *file* operand, the *mkfifo* utility will perform actions equivalent to the **XSH** specification  
19360 *mkfifo()* function, called with the following arguments:

- 19361 1. The *file* operand is used as the *path* argument.
- 19362 2. The value of the bitwise inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP, S\_IWGRP,  
19363 S\_IROTH and S\_IWOTH is used as the *mode* argument. (If the **-m** option is specified, the  
19364 *mode* option-argument overrides this default.)

19365 **OPTIONS**19366 The *mkfifo* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

19367 The following option is supported:

19368 **-m** *mode*

19369 Set the file permission bits of the newly-created FIFO to the specified *mode* value. The  
19370 *mode* option-argument will be the same as the *mode* operand defined for the *chmod*  
19371 utility. In the *symbolic\_mode* strings, the *op* characters "+" and "-" will be interpreted  
19372 relative to an assumed initial mode of a=rw.

19373 **OPERANDS**

19374 The following operand is supported:

19375 *file* A pathname of the FIFO special file to be created.19376 **STDIN**

19377 Not used.

19378 **INPUT FILES**

19379 None.

19380 **ENVIRONMENT VARIABLES**19381 The following environment variables affect the execution of *mkfifo*:

19382 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
19383 **LANG** is unset or null, the corresponding value from the implementation-dependent  
19384 default locale will be used. If any of the internationalisation variables contains an  
19385 invalid setting, the utility will behave as if none of the variables had been defined.

19386 **LC\_ALL**

19387 If set to a non-empty string value, override the values of all the other  
19388 internationalisation variables.

19389 **LC\_CTYPE**

19390 Determine the locale for the interpretation of sequences of bytes of text data as  
19391 characters (for example, single- as opposed to multi-byte characters in arguments).

19392 **LC\_MESSAGES**

19393 Determine the locale that should be used to affect the format and contents of diagnostic  
19394 messages written to standard error.

19395 EX **NLSPATH**  
19396 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

19397 **ASYNCHRONOUS EVENTS**  
19398 Default.

19399 **STDOUT**  
19400 Not used.

19401 **STDERR**  
19402 Used only for diagnostic messages.

19403 **OUTPUT FILES**  
19404 None.

19405 **EXTENDED DESCRIPTION**  
19406 None.

19407 **EXIT STATUS**  
19408 The following exit values are returned:  
19409 0 All the specified FIFO special files were created successfully.  
19410 >0 An error occurred.

19411 **CONSEQUENCES OF ERRORS**  
19412 Default.

19413 **APPLICATION USAGE**  
19414 None.

19415 **EXAMPLES**  
19416 None.

19417 **FUTURE DIRECTIONS**  
19418 None.

19419 **SEE ALSO**  
19420 *umask*, the **XSH** specification description of *mkfifo()*.

19421 **CHANGE HISTORY**  
19422 First released in Issue 3.

19423 **Issue 4**  
19424 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 19425 NAME

19426       more — display files on a page-by-page basis

## 19427 SYNOPSIS

19428       more [-ceisu][-n *number*][-p *command*][-t *tagstring*][*file...*]

19429 OB       more [-ceisu][-n *number*][+*command*][-t *tagstring*][*file...*]

## 19430 DESCRIPTION

19431       The *more* utility reads files and either writes them to the terminal on a page-by-page basis or  
 19432 filters them to standard output. If standard output is not a terminal device, all input files are  
 19433 copied to standard output in their entirety, without modification. If standard output is a  
 19434 terminal device, the files will be written a number of lines (one screenful) at a time under the  
 19435 control of user commands; see the EXTENDED DESCRIPTION section.

19436       Certain block-mode terminals do not have all the capabilities necessary to support the complete  
 19437 *more* definition; they are incapable of accepting commands that are not terminated with a  
 19438 newline character. Implementations that support such terminals provide an operating mode to  
 19439 *more* in which all commands can be terminated with a newline character on those terminals.  
 19440 This mode will:

- 19441       • be documented in the system documentation
- 19442       • at invocation, inform the user of the terminal deficiency that requires the newline character  
 19443 usage and provide instructions on how this warning can be suppressed in future invocations
- 19444       • not be required for implementations supporting only fully capable terminals
- 19445       • not affect commands already requiring newline characters
- 19446       • not affect users on the capable terminals from using *more* as described in this specification

## 19447 OPTIONS

19448 OB       The *more* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 19449 that +*command* of the obsolescent version uses a non-standard syntax.

19450       The following options are supported:

19451       -c       If a screen is to be written that has no lines in common with the current screen, or *more*  
 19452 is writing its first screen, *more* will not scroll the screen, but instead will redraw each  
 19453 line of the screen in turn, from the top of the screen to the bottom. In addition, if *more* is  
 19454 writing its first screen, the screen will be cleared. This option may be ignored on  
 19455 devices that do not have the ability to clear to the end of a line or the end of a screen.

19456       -e       Exit immediately after writing the last line of the last file in the argument list; see the  
 19457 EXTENDED DESCRIPTION section.

19458       -i       Perform pattern matching in searches without regard to case. See the XBD  
 19459 specification, **Section 7.2, Regular Expression General Requirements**.

19460       -n *number*

19461       Specify the number of lines per screenful. The *number* argument is a positive decimal  
 19462 integer. The -n option overrides any values obtained from the environment.

19463       -p *command*

19464 OB       +*command*

19465       For each file examined, initially execute the *more* command in the *command* argument.  
 19466 If the command is a positioning command, such as a line number or a regular  
 19467 expression search, set the current position (see the EXTENDED DESCRIPTION section)  
 19468 to represent the final results of the command, without writing any intermediate lines of



19469 the file. For example, the two commands:

19470 `more -p 1000j file`

19471 `more -p 1000G file`

19472 would be equivalent and start the display with the current position at line 1000,  
19473 bypassing the lines that **j** would write and scroll off the screen if it had been issued  
19474 during the file examination. If the positioning command is unsuccessful, the first line  
19475 in the file will be the current position.

19476 **-s** Replace consecutive empty lines with a single empty line.

19477 **-t tagstring**

19478 Write the screenful of the file containing the tag named by the *tagstring* argument. See  
19479 the *ctags* utility. The tags feature represented by **-t tagstring** and the **:t** command is  
19480 optional. It is provided on any system that also provides a conforming implementation  
19481 of *ctags*; otherwise, the use of **-t** produces undefined results.

19482 **-u** Treat a backspace character as a printable control character, displayed as an  
19483 implementation-dependent character sequence (see the EXTENDED DESCRIPTION  
19484 section), suppressing backspacing and the special handling that produces underlined or  
19485 standout-mode text on some terminal types. Also, do not ignore a carriage-return  
19486 character at the end of a line.

19487 **OB** If both the **-t tagstring** and **-p command** (or the obsolescent *+command*) options are given, the  
19488 **-t tagstring** will be processed first; that is, the file containing the tag is selected by **-t** and then  
19489 the command is executed.

## 19490 OPERANDS

19491 The following operand is supported:

19492 **file** A pathname of an input file. If no *file* operands are specified, the standard input will be  
19493 used. If a *file* is "-", the standard input will be read at that point in the sequence.

## 19494 STDIN

19495 The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".

## 19496 INPUT FILES

19497 The input files being examined must be text files. If standard output is a terminal, standard  
19498 error will be used to read commands from the user. If standard output is a terminal, standard  
19499 error is not readable, and command input is needed, *more* will terminate with an error indicating  
19500 that it was unable to read user commands. If standard output is not a terminal, no error will  
19501 result if standard error cannot be opened for reading.

## 19502 ENVIRONMENT VARIABLES

19503 The following environment variables affect the execution of *more*:

### 19504 COLUMNS

19505 Override the system-selected horizontal screen size. See the **XBD** specification,  
19506 **Chapter 6, Environment Variables** for valid values and results when it is unset or null.

### 19507 EDITOR

19508 Used by the **v** command to select an editor; see the EXTENDED DESCRIPTION section.

19509 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
19510 **LANG** is unset or null, the corresponding value from the implementation-dependent  
19511 default locale will be used. If any of the internationalisation variables contains an  
19512 invalid setting, the utility will behave as if none of the variables had been defined.

19513 **LC\_ALL**  
 19514 If set to a non-empty string value, override the values of all the other  
 19515 internationalisation variables.

19516 **LC\_COLLATE**  
 19517 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 19518 character collating elements within regular expressions.

19519 **LC\_CTYPE**  
 19520 Determine the locale for the interpretation of sequences of bytes of text data as  
 19521 characters (for example, single- as opposed to multi-byte characters in arguments and  
 19522 input files) and the behaviour of character classes within regular expressions.

19523 **LC\_MESSAGES**  
 19524 Determine the locale that should be used to affect the format and contents of diagnostic  
 19525 messages written to standard error and informative messages written to standard  
 19526 output.

19527 EX **NLSPATH**  
 19528 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

19529 **LINES** Override the system-selected vertical screen size, used as the number of lines in a  
 19530 screenful. See the **XBD** specification, **Chapter 6, Environment Variables** for valid  
 19531 values and results when it is unset or null. The **-n** option takes precedence over the  
 19532 **LINES** variable for determining the number of lines in a screenful.

19533 **MORE** Determine a string containing options described in the **OPTIONS** section preceded  
 19534 with hyphens and blank-character-separated as on the command line. Any command-  
 19535 line options are processed after those in the **MORE** variable, as if the command line  
 19536 were:  
 19537 `more $MORE options operands`

19538 The **MORE** variable takes precedence over the **TERM** and **LINES** variables for  
 19539 determining the number of lines in a screenful.

19540 **TERM** Determine the name of the terminal type. If this variable is unset or null, an  
 19541 unspecified default terminal type will be used.

19542 **ASYNCHRONOUS EVENTS**  
 19543 Default.

19544 **STDOUT**  
 19545 The standard output will be used to write the contents of the input files.

19546 **STDERR**  
 19547 Used for diagnostic messages and user commands (see the **INPUT FILES** section), and, if  
 19548 standard output is a terminal device, to write a prompting string. The prompting string will  
 19549 only appear as the last line of the screen and will contain the name of the file currently being  
 19550 examined, but it is otherwise unspecified. User input for the **/**, **?**, **:e** and **:t** commands will be  
 19551 written on the same line of the screen as the prompt. It is unspecified if informational messages  
 19552 are written for other user commands.

19553 **OUTPUT FILES**  
 19554 None.

19555 **EXTENDED DESCRIPTION**  
 19556 The number of lines available per screen is determined by the **-n** option, if present or by  
 19557 examining values in the environment (see the **ENVIRONMENT VARIABLES** section). If neither

19558 method yields a number, an unspecified number of lines will be used. The actual number of  
 19559 lines written will be one less than this number, as the last line of the screen will be used to write  
 19560 a user prompt and user input. If the number of lines available per screen is less than four, the  
 19561 results are undefined.

19562 In the following descriptions, the *current position* refers to two things:

- 19563 • the position of the current line on the screen
- 19564 • the line number (in the file) of the current line on the screen.

19565 Usually, the line on the screen corresponding to the current position is the third line on the  
 19566 screen. If this is not possible (there are fewer than three lines to display or this is the first page of  
 19567 the file, or it is the last page of the file), then the current position is either the first or last line on  
 19568 the screen as described later.

19569 The number of columns available per line is determined by examining values in the  
 19570 environment (see the ENVIRONMENT VARIABLES section), with a default value as described  
 19571 in the **XBD specification, Chapter 6, Environment Variables**. The *more* utility writes lines  
 19572 containing more characters than would fit into this number of columns by breaking the line into  
 19573 one or more logical lines where each of these lines but the last contains the number of characters  
 19574 needed to fill the columns. The logical lines are written independently of each other; that is,  
 19575 commands affecting a single line affect them separately.

19576 When standard output is a terminal and **-u** is not specified, *more* treats backspace characters and  
 19577 carriage-return characters specially:

- 19578 • A character, followed first by a backspace character, then by an underscore (**\_**), will cause  
 19579 that character to be written as underlined text, if the terminal type supports that. An  
 19580 underscore, followed first by a backspace character, then any character, will also cause that  
 19581 character to be written as underlined text, if the terminal type supports that.
- 19582 • A backspace character that appears between two identical printable characters will cause the  
 19583 first of those two characters to be written as emboldened text (that is, visually brighter,  
 19584 standout mode or inverse-video mode), if the terminal type supports that, and the second to  
 19585 be discarded. Immediately subsequent occurrences of backspaces/character pairs for that  
 19586 same character will also be discarded. (For example, the sequence **a\ba\ba\ba** is interpreted  
 19587 as a single emboldened **a**.)
- 19588 • Other backspace character sequences will be written directly to the terminal, which generally  
 19589 cause the character preceding the backspace character to be suppressed in the display.
- 19590 • A carriage-return character at the end of a line will be ignored, rather than being written as a  
 19591 control character, as described in the next paragraph.

19592 It is implementation-dependent how other non-printable characters are written.  
 19593 Implementations should use the same format that they use for the **ex print** command.

19594 If the **-t** option was specified, *more* will write a screen of the file containing the specified tag in  
 19595 the current position.

19596 If the **-p** option was specified, *more* will execute the command supplied in the *command* option-  
 19597 argument, and write the screen so that the current position is the current position that would  
 19598 result from applying that command to the input file.

19599 If neither the **-p** or **-t** options were specified, *more* will write the first screen of the first input file.

19600 Once the initial screen has been written, *more* will prompt the user and, based on the interactive  
 19601 user input, will modify the screen or exit. If the modification of the screen results in a screen that  
 19602 has lines in common with the current screen, *more* will scroll the screen rather than clearing and

19603 redrawing the screen (unless the `-c` option is specified). If the modification of the screen results  
 19604 in a screen that has no lines in common with the current screen, *more* will clear and redraw the  
 19605 screen.

19606 If the standard output is not a terminal device, *more* will always exit when it reaches end-of-file  
 19607 on the last file in its argument list. Otherwise, for all files but the last, *more* will prompt, with an  
 19608 indication that it has reached the end-of-file, along with the name of the next file. For the last file  
 19609 specified, or for the standard input if no file is specified, *more* will prompt, indicating end-of-file,  
 19610 and accept additional commands. If the next command specifies forward scrolling, *more* will  
 19611 exit. If the `-e` option is specified, *more* will exit immediately after writing the last line of the last  
 19612 file.

19613 Several of the commands described in this section move the current position backwards in the  
 19614 input stream. In the case that text is being taken from a non-rewindable stream, such as a pipe,  
 19615 it is implementation-dependent how much backward motion is supported.

19616 If a scrolling command cannot be performed because there are insufficient lines to scroll, *more*  
 19617 will alert the terminal. The scrolling commands are `b`, `<control>-B`, `d`, `<control>-D`, `f`, `<control>-F`,  
 19618 `g`, `G`, `j`, `k`, `s`, `u` and `<control>-U`.

19619 The interactive commands in the following sections are supported. Some commands can be  
 19620 preceded by a decimal integer, called *count* in the following descriptions. If not specified with  
 19621 the command, *count* defaults to 1.

19622 In the following descriptions, *pattern* is a basic regular expression, as described in the **XBD**  
 19623 specification, **Section 7.3, Basic Regular Expressions**. The term “examine” is historical usage  
 19624 meaning “open the file for viewing”; for example, *more foo* would be expressed as examining  
 19625 file **foo**.

## 19626 **Help**

19627 *Synopsis:*     `h`

19628 Write a summary of these commands and other implementation-dependent commands.

## 19629 **Move Forward One Screenful**

19630 *Synopsis:*     `[count]f`

19631 *Synopsis:*     `[count]<control>-F`

19632 Move forward *count* lines, with a default of one screenful. At end-of-file, *more* will continue with  
 19633 the next file in the list, or exit if the current file is the last file in the list.

## 19634 **Move Backward One Screenful**

19635 *Synopsis:*     `[count]b`

19636 *Synopsis:*     `[count]<control>-B`

19637 Move backward *count* lines, with a default of one screenful (see option `-n`). If *count* is more than  
 19638 the screen size, only the final screenful will be written.

### 19639 **Scroll Forward One Line**

19640 *Synopsis:* [ *count* ]<space>  
 19641 *Synopsis:* [ *count* ]j  
 19642 *Synopsis:* [ *count* ]<newline>

19643 Scroll forward *count* lines. The default *count* for the space character will be one screenful; for j  
 19644 and newline character, one line. The entire *count* lines will be written, even if *count* is more than  
 19645 the screen size. At end-of-file, a newline character will cause *more* to continue with the next file  
 19646 in the list, or exit if the current file is the last file in the list.

### 19647 **Scroll Backward One Line**

19648 *Synopsis:* [ *count* ]k

19649 Scroll backward *count* lines, with a default of 1. The entire *count* lines will be written, even if  
 19650 *count* is more than the screen size.

### 19651 **Scroll Forward One Half Screenful**

19652 *Synopsis:* [ *count* ]d  
 19653 *Synopsis:* [ *count* ]<control>-D

19654 Scroll forward *count* lines, with a default of one half of the screen size. If *count* is specified, it will  
 19655 become the new default for subsequent **d** and **u** commands.

### 19656 **Skip Forward One Line**

19657 *Synopsis:* [ *count* ]s

19658 Skip forward *count* lines, with a default of 1, and write the next screenful beginning at that point.  
 19659 If *count* would cause the current position to be such that less than one screenful would be  
 19660 written, the last screenful in the file will be written.

### 19661 **Scroll Backward One Half Screenful**

19662 *Synopsis:* [ *count* ]u  
 19663 *Synopsis:* [ *count* ]<control>-U

19664 Scroll backward *count* lines, with a default of one half of the screen size. If *count* is specified, it  
 19665 will become the new default for subsequent **d** and **u** commands.

### 19666 **Go to Beginning of File**

19667 *Synopsis:* [ *count* ]g

19668 Go to line *count* in the file, with a default of 1 (beginning of file). Scroll or rewrite the screen so  
 19669 that that line is at the current position. |

### 19670 **Go to End-of-file**

19671 *Synopsis:* [ *count* ]G

19672 Go to line *count* in the file, with a default of the end of the file. If no *count* is specified, scroll or  
 19673 rewrite the screen so that the last line in the file is at the bottom of the screen. If *count* is  
 19674 specified, scroll or rewrite the screen so that that line is at the current position. |

### 19675 Refresh the Screen

19676 *Synopsis:*     `r`  
 19677 *Synopsis:*     `<control>-L`

19678 Refresh the screen.

### 19679 Discard and Refresh

19680 *Synopsis:*     `R`

19681 Refresh the screen, discarding any buffered input. If the current file is non-seekable, buffered  
 19682 input will not be discarded and the **R** command is equivalent to the **r** command.

### 19683 Mark Position

19684 *Synopsis:*     `mletter`

19685 Mark the current position with the letter named by *letter*, where *letter* represents the name of one  
 19686 of the lower-case letters of the portable character set. When a new file is examined, all marks  
 19687 may be lost.

### 19688 Return to Mark

19689 *Synopsis:*     `'letter`

19690 Return to the position that was previously marked with the letter named by *letter*, making that  
 19691 line the current position.

### 19692 Return to Previous Position

19693 *Synopsis:*     `' '`

19694 Return to the position from which the last large movement command was executed (where a  
 19695 “large movement” is defined as any movement of more than a screenful of lines). If no such  
 19696 movements have been made, return to the beginning of the file.

### 19697 Search Forward for Pattern

19698 *Synopsis:*     `[count]/[!]pattern<newline>`

19699 Search forward in the file for the *count*th line containing the *pattern*. The *count* defaults to 1. The  
 19700 search will start at the line following the current position. If the search is successful, the screen  
 19701 will be modified so that the searched-for line is in the current position. The null regular  
 19702 expression (`/` followed by a newline character) will repeat the search using the previous regular  
 19703 expression. If the character `!` is included, the lines for searching will be those that do not  
 19704 contain the *pattern*.

19705 If a match is found for the *pattern*, the logical line containing the pattern will be written in the  
 19706 current position. If the matching line is already on the screen, the screen will be scrolled to make  
 19707 that line the current position; otherwise, a full screen will be written. If no match is found for the  
 19708 pattern, a message to that effect will be written as a part of the prompt, and the screen and the  
 19709 current position will remain unchanged. However, if no match is found and the input is the  
 19710 standard input, the screen may be scrolled to the last screenful of the input.

19711 **Search Backward for Pattern**19712 *Synopsis:*        `[count]?[!]pattern<newline>`

19713 Search backward in the file for the *countth* line containing the *pattern*. The search will start at the  
 19714 line immediately before the current position. If the search is successful, the screen will be  
 19715 modified so that the searched-for line is in the current position. The null regular expression (?  
 19716 followed by a newline character) will repeat the search using the previous regular expression. If  
 19717 the character "!" is included, the lines for searching will be those that do not contain the *pattern*.

19718 If a match is found for the *pattern*, the logical line containing the pattern will be written in the  
 19719 current position. If the matching line is already on the screen, the screen will be scrolled to make  
 19720 that line the current position; otherwise, a full screen will be written. If no match is found for the  
 19721 pattern, a message to that effect will be written as a part of the prompt, and the screen and the  
 19722 current position will remain unchanged. However, if no match is found and the input is the  
 19723 standard input, the screen may be scrolled to the last screenful of the input.

19724 **Repeat Search**19725 *Synopsis:*        `[count]n`

19726 Repeat the previous search for *countth* line (default 1) containing the last *pattern* (or not  
 19727 containing the last *pattern*, if the previous search was `/!` or `?!`).

19728 **Repeat Search in Reverse**19729 *Synopsis:*        `[count]N`

19730 Repeat the search in the opposite direction of the previous search for the *countth* line (default 1)  
 19731 containing the last *pattern* (or not containing the last *pattern*, if the previous search was `/!` or `?!`).

19732 **Examine New File**19733 *Synopsis:*        `:e [filename]<newline>`

19734 Examine a new file. If the *filename* argument is not specified, the current file (see the `:n` and `:p`  
 19735 commands below) from the list of files in the command line will be reexamined. The *filename*  
 19736 will be subjected to the process of shell word expansions (see Section 2.6 on page 31); if more  
 19737 than a single pathname results, the effects are unspecified. If *filename* is a number sign (#), the  
 19738 previously examined file will be reexamined. If *filename* refers to a non-seekable file, the results  
 19739 are unspecified.

19740 **Examine Next File**19741 *Synopsis:*        `[count]:n`

19742 Examine the next file. If a number *count* is specified, the *countth* next file will be examined. If  
 19743 *filename* refers to a non-seekable file, the results are unspecified.

19744 **Examine Previous File**19745 *Synopsis:*        `[count]:p`

19746 Examine the previous file. If a number *count* is specified, the *countth* previous file will be  
 19747 examined. If *filename* refers to a non-seekable file, the results are unspecified.

19748 **Go to Tag**19749 *Synopsis:*     :t *tagstring*<newline>

19750 Go to the supplied *tagstring* and scroll/rewrite the screen with that line in the current position;  
 19751 see the -t option. If the *ctags* utility is not supported by the system, the use of :t produces  
 19752 undefined results.

19753 **Invoke Editor**19754 *Synopsis:*     v

19755 Invoke an editor to edit the current file being examined. If standard input is being examined, the  
 19756 results are unspecified. The name of the editor will be taken from the environment variable  
 19757 *EDITOR*, or defaults to *vi*. If *EDITOR* represents either *vi* or *ex*, the editor will be invoked with  
 19758 options such that the current editor line is the physical line corresponding to the current position  
 19759 in *more* at the time of invocation. For example, either *ex* or *vi* is invoked by specifying the editor  
 19760 name and following that with -c *linenumber*. It is implementation-dependent whether line-  
 19761 setting options are passed to editors other than *vi* and *ex*.

19762 The file types that can be edited are implementation-dependent.

19763 When the editor exits, *more* will resume on the current file by rewriting the screen with the  
 19764 current line as the current position.

19765 **Display Position**19766 *Synopsis:*     =19767 *Synopsis:*     <control>-G

19768 Write the name of the file currently being examined, the number relative to the total number of  
 19769 files there are to examine, the current line number, the current byte number and the total bytes to  
 19770 write and what percentage of the file precedes the current position. If *more* is reading from  
 19771 standard input, or the file is shorter than a single screen, some of these items need not be written.  
 19772 All of these items will reference the first byte of the line after the last line written.

19773 **Quit**19774 *Synopsis:*     q19775 *Synopsis:*     :q19776 *Synopsis:*     ZZ19777 Exit *more*.19778 **EXIT STATUS**

19779 The following exit values are returned:

19780     0 Successful completion.

19781     &gt;0 An error occurred.

19782 **CONSEQUENCES OF ERRORS**

19783 If an error is encountered accessing a file when using the :n command, *more* will attempt to  
 19784 examine the next file in the argument list, but the final exit status will be affected. If an error is  
 19785 encountered accessing a file via the :p command, *more* will attempt to examine the previous file  
 19786 in the argument list, but the final exit status will be affected. If an error is encountered accessing  
 19787 a file via the :e command, *more* will remain in the current file and the final exit status will not be  
 19788 affected.



**19789 APPLICATION USAGE**

19790 The operating mode referred to for block-mode terminals effectively adds a newline character to  
19791 each synopsis line that currently has none. So, for example, **d**<newline> would page one  
19792 screenful. The mode could be triggered by a command-line option, environment variable or  
19793 some other method.

19794 When the standard output is not a terminal, none of the filter-modification options are effective.  
19795 This is based on historical practice. For example, a typical implementation of *man* pipes its  
19796 output through *more -s* to squeeze excess white space for terminal users. When *man* is piped to  
19797 *lp*, however, it is undesirable for this squeezing to happen.

**19798 EXAMPLES**

19799 The **-p** allows arbitrary commands to be executed at the start of each file. Examples are:

19800 *more -p G file1 file2*

19801 Examine each file starting with its last screenful.

19802 *more -p 100 file1 file2*

19803 Examine each file starting with line 100 in the current position (usually the third line, so  
19804 line 98 would be the first line written).

19805 *more -p /100 file1 file2*

19806 Examine each file starting with the first line containing the string 100 in the current  
19807 position

**19808 FUTURE DIRECTIONS**

19809 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
19810 interface definition to the IEEE PASC 1003.2 Interpretations Committee which is identifying the  
19811 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
19812 finalised.

**19813 SEE ALSO**

19814 None.

**19815 CHANGE HISTORY**

19816 First released in Issue 4.

**19817 Issue 5**

19818 FUTURE DIRECTIONS section added.

## 19819 NAME

19820 mv — move files

## 19821 SYNOPSIS

19822 mv [-fi] *source\_file target\_file*19823 mv [-fi] *source\_file... target\_file*

## 19824 DESCRIPTION

19825 In the first synopsis form, the *mv* utility moves the file named by the *source\_file* operand to the  
 19826 *destination* specified by the *target\_file*. This first synopsis form is assumed when the final  
 19827 operand does not name an existing directory.

19828 In the second synopsis form, *mv* moves each file named by a *source\_file* operand to a *destination*  
 19829 file in the existing directory named by the *target\_dir* operand. The *destination* path for each  
 19830 *source\_file* is the concatenation of the target directory, a single slash character, and the last  
 19831 pathname component of the *source\_file*. This second form is assumed when the final operand  
 19832 names an existing directory.

19833 If any operand specifies an existing file of a type not specified by the **XSH** specification, the  
 19834 behaviour is implementation-dependent.

19835 For each *source\_file* the following steps are taken:

19836 1. If the destination path exists, the **-f** option is not specified, and either of the following  
 19837 conditions is true:

19838 a. The permissions of the destination path do not permit writing and the standard input  
 19839 is a terminal.

19840 b. The **-i** option is specified.

19841 The *mv* utility will write a prompt to standard error and read a line from standard input. If  
 19842 the response is not affirmative, *mv* will do nothing more with the current *source\_file* and go  
 19843 on to any remaining *source\_files*.

19844 2. The *mv* utility will perform actions equivalent to the **XSH** specification *rename()* function,  
 19845 called with the following arguments:

19846 a. The *source\_file* operand is used as the *old* argument.

19847 b. The destination path is used as the *new* argument.

19848 If this succeeds, *mv* will do nothing more with the current *source\_file* and go on to any  
 19849 remaining *source\_files*. If this fails for any reasons other than those described for the *errno*  
 19850 [EXDEV] in the **XSH** specification, *mv* will write a diagnostic message to standard error, do  
 19851 nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

19852 3. If the destination path exists, and it is a file of type directory and *source\_file* is not a file of  
 19853 type directory, or it is a file not of type directory and *source\_file* is a file of type directory,  
 19854 *mv* will write a diagnostic message to standard error, do nothing more with the current  
 19855 *source\_file*, and go on to any remaining *source\_files*.

19856 4. If the destination path exists, *mv* will attempt to remove it. If this fails for any reason, *mv*  
 19857 will write a diagnostic message to standard error, do nothing more with the current  
 19858 *source\_file*, and go on to any remaining *source\_files*.

19859 5. The file hierarchy rooted in *source\_file* will be duplicated as a file hierarchy rooted in the  
 19860 destination path. The following characteristics of each file in the file hierarchy will be  
 19861 duplicated:

- 19862 • the time of last data modification and time of last access
- 19863 • the user ID and group ID
- 19864 • the file mode.

19865 If the user ID, group ID or file mode of a regular file cannot be duplicated, the file mode  
 19866 bits S\_ISUID and S\_ISGID will not be duplicated.

19867 When files are duplicated to another file system, the implementation may require that the  
 19868 process invoking *mv* has read access to each file being duplicated.

19869 If the duplication of the file hierarchy fails for any reason, *mv* will write a diagnostic  
 19870 message to standard error, do nothing more with the current *source\_file*, and go on to any  
 19871 remaining *source\_files*.

19872 If the duplication of the file characteristics fails for any reason, *mv* will write a diagnostic  
 19873 message to standard error, but this failure will not cause *mv* to modify its exit status.

19874 6. The file hierarchy rooted in *source\_file* will be removed. If this fails for any reason, *mv* will  
 19875 write a diagnostic message to the standard error, do nothing more with the current  
 19876 *source\_file*, and go on to any remaining *source\_files*.

#### 19877 OPTIONS

19878 The *mv* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

19879 The following options are supported:

19880 **-f** Do not prompt for confirmation if the destination path exists. Any previous  
 19881 occurrences of the **-i** option will be ignored.

19882 **-i** Prompt for confirmation if the destination path exists. Any previous occurrences of the  
 19883 **-f** option will be ignored.

19884 Specifying more than one of the **-f** or **-i** options is not considered an error. The last option  
 19885 specified will determine the behaviour of *mv*.

#### 19886 OPERANDS

19887 The following operands are supported:

19888 *source\_file*  
 19889 A pathname of a file or directory to be moved.

19890 *target\_file*  
 19891 A new pathname for the file or directory being moved.

19892 *target\_dir*  
 19893 A pathname of an existing directory into which to move the input files.

#### 19894 STDIN

19895 Used to read an input line in response to each prompt specified in the **STDERR** section.  
 19896 Otherwise, the standard input will not be used.

#### 19897 INPUT FILES

19898 The input files specified by each *source\_file* operand can be of any file type.

#### 19899 ENVIRONMENT VARIABLES

19900 The following environment variables affect the execution of *mv*:

|          |                               |                                                                                                                |
|----------|-------------------------------|----------------------------------------------------------------------------------------------------------------|
| 19901    | <b>LANG</b>                   | Provide a default value for the internationalisation variables that are unset or null. If                      |
| 19902    |                               | <i>LANG</i> is unset or null, the corresponding value from the implementation-dependent                        |
| 19903    |                               | default locale will be used. If any of the internationalisation variables contains an                          |
| 19904    |                               | invalid setting, the utility will behave as if none of the variables had been defined.                         |
| 19905    | <b>LC_ALL</b>                 |                                                                                                                |
| 19906    |                               | If set to a non-empty string value, override the values of all the other                                       |
| 19907    |                               | internationalisation variables.                                                                                |
| 19908    | <b>LC_COLLATE</b>             |                                                                                                                |
| 19909    |                               | Determine the locale for the behaviour of ranges, equivalence classes and multi-                               |
| 19910    |                               | character collating elements used in the extended regular expression defined for the                           |
| 19911    |                               | <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category.                                              |
| 19912    | <b>LC_CTYPE</b>               |                                                                                                                |
| 19913    |                               | Determine the locale for the interpretation of sequences of bytes of text data as                              |
| 19914    |                               | characters (for example, single- as opposed to multi-byte characters in arguments and                          |
| 19915    |                               | input files), the behaviour of character classes used in the extended regular expression                       |
| 19916    |                               | defined for the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category.                              |
| 19917    | <b>LC_MESSAGES</b>            |                                                                                                                |
| 19918    |                               | Determine the locale for the processing of affirmative responses that should be used to                        |
| 19919    |                               | affect the format and contents of diagnostic messages written to standard error.                               |
| 19920 EX | <b>NLSPATH</b>                |                                                                                                                |
| 19921    |                               | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                        |
| 19922    | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                |
| 19923    |                               | Default.                                                                                                       |
| 19924    | <b>STDOUT</b>                 |                                                                                                                |
| 19925    |                               | Not used.                                                                                                      |
| 19926    | <b>STDERR</b>                 |                                                                                                                |
| 19927    |                               | Prompts will be written to the standard error under the conditions specified in the                            |
| 19928    |                               | DESCRIPTION section. The prompts will contain the <i>destination</i> pathname, but their format is             |
| 19929    |                               | otherwise unspecified. Otherwise, the standard error will be used only for diagnostic messages.                |
| 19930    | <b>OUTPUT FILES</b>           |                                                                                                                |
| 19931    |                               | The output files may be of any file type.                                                                      |
| 19932    | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                |
| 19933    |                               | None.                                                                                                          |
| 19934    | <b>EXIT STATUS</b>            |                                                                                                                |
| 19935    |                               | The following exit values are returned:                                                                        |
| 19936    | 0                             | All input files were moved successfully.                                                                       |
| 19937    | >0                            | An error occurred.                                                                                             |
| 19938    | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                |
| 19939    |                               | If the copying or removal of <i>source_file</i> is prematurely terminated by a signal or error, <i>mv</i> may  |
| 19940    |                               | leave a partial copy of <i>source_file</i> at the source or destination. The <i>mv</i> utility will not modify |
| 19941    |                               | both <i>source_file</i> and the destination path simultaneously; termination at any point will leave           |
| 19942    |                               | either <i>source_file</i> or the destination path complete.                                                    |
| 19943    | <b>APPLICATION USAGE</b>      |                                                                                                                |
| 19944    |                               | None.                                                                                                          |

19945 **EXAMPLES**

19946 If the current directory contains only files **a** (of any type defined by the **XSH** specification), **b**  
19947 (also of any type), and a directory **c**:

19948 mv a b c

19949 mv c d

19950 will result with the original files **a** and **b** residing in the directory **d** in the current directory.

19951 **FUTURE DIRECTIONS**

19952 None.

19953 **SEE ALSO**

19954 *cp*, *ln*.

19955 **CHANGE HISTORY**

19956 First released in Issue 2.

19957 **Issue 4**

19958 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 19959 NAME

19960 newgrp — change to a new group

## 19961 SYNOPSIS

19962 newgrp [-l][group]

19963 OB newgrp [-][group]

## 19964 DESCRIPTION

19965 The *newgrp* utility creates a new shell execution environment with a new real and effective  
 19966 group identification. Of the attributes listed in Section 2.12 on page 63, the new shell execution  
 19967 environment will retain the working directory, file creation mask and exported variables from  
 19968 the previous environment (that is, open files, traps, unexported variables, alias definitions, shell  
 19969 functions and *set* options may be lost). All other aspects of the process environment that are  
 19970 preserved by the *exec* family of functions in the **XSH** specification also are preserved by *newgrp*;  
 19971 whether other aspects are preserved is unspecified.

19972 A failure to assign the new group identifications (for example, for security or password-related  
 19973 reasons) does not prevent the new shell execution environment from being created.

19974 FIPS The *newgrp* utility affects the supplemental groups for the process as follows:

19975 • On systems where the effective group ID is normally in the supplementary group list (or  
 19976 whenever the old effective group ID actually is in the supplementary group list):

19977 — If the new effective group ID is also in the supplementary group list, *newgrp* will change  
 19978 the effective group ID.

19979 — If the new effective group ID is not in the supplementary group list, *newgrp* will add the  
 19980 new effective group ID to the list, if there is room to add it.

19981 • On systems where the effective group ID is not normally in the supplementary group list (or  
 19982 whenever the old effective group ID is not in the supplementary group list):

19983 — If the new effective group ID is in the supplementary group list, *newgrp* will delete it.

19984 — If the old effective group ID is not in the supplementary list, *newgrp* will add it if there is  
 19985 room.

19986 **Note:** The **XSH** specification does not specify whether the effective group ID of a process is  
 19987 included in its supplementary group list.

19988 With no operands, *newgrp* will change the effective group back to the groups identified in the  
 19989 FIPS user's user entry, and will set the list of supplementary groups to that set in the user's group  
 19990 database entries.

19991 If a password is required for the specified group, and the user is not listed as a member of that  
 19992 group in the group database, the user will be prompted to enter the correct password for that  
 19993 group. If the user is listed as a member of that group, no password will be requested. If no  
 19994 password is required for the specified group, it is implementation-dependent whether users not  
 19995 listed as members of that group can change to that group. Whether or not a password is  
 19996 required, implementation-dependent system accounting or security mechanisms may impose  
 19997 additional authorisation restrictions that may cause *newgrp* to write a diagnostic message and  
 19998 suppress the changing of the group identification.

## 19999 OPTIONS

20000 The *newgrp* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**,  
 20001 OB except that the obsolescent version uses "-" in a non-standard manner.

20002 The following option is supported:

20003 **-l** (The letter ell.)

20004 **OB** **-** Change the environment to what would be expected if the user actually logged in  
20005 again.

## 20006 OPERANDS

20007 The following operand is supported:

20008 *group* A group name from the group database or a non-negative numeric group ID. Specifies  
20009 the group ID to which the real and effective group IDs will be set. If *group* is a non-  
20010 negative numeric string and exists in the group database as a group name (see  
20011 *getgrnam()*), the numeric group ID associated with that group name will be used as the  
20012 group ID.

## 20013 STDIN

20014 Not used.

## 20015 INPUT FILES

20016 The file **/dev/tty** is used to read a single line of text for password checking, when one is required.

## 20017 ENVIRONMENT VARIABLES

20018 The following environment variables affect the execution of *newgrp*:

20019 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
20020 **LANG** is unset or null, the corresponding value from the implementation-dependent  
20021 default locale will be used. If any of the internationalisation variables contains an  
20022 invalid setting, the utility will behave as if none of the variables had been defined.

20023 **LC\_ALL**

20024 If set to a non-empty string value, override the values of all the other  
20025 internationalisation variables.

20026 **LC\_CTYPE**

20027 Determine the locale for the interpretation of sequences of bytes of text data as  
20028 characters (for example, single- as opposed to multi-byte characters in arguments).

20029 **LC\_MESSAGES**

20030 Determine the locale that should be used to affect the format and contents of diagnostic  
20031 messages written to standard error.

20032 **EX** **NLSPATH**

20033 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

## 20034 ASYNCHRONOUS EVENTS

20035 Default.

## 20036 STDOUT

20037 Not used.

## 20038 STDERR

20039 Used for diagnostic messages and a prompt string for a password, if one is required. Diagnostic  
20040 messages may be written in cases where the exit status is not available; see **EXIT STATUS**.

## 20041 OUTPUT FILES

20042 None.

## 20043 EXTENDED DESCRIPTION

20044 None.

20045 **EXIT STATUS**

20046 If *newgrp* succeeds in creating a new shell execution environment, whether or not the group  
 20047 identification was changed successfully, the exit status will be the exit status of the shell.  
 20048 Otherwise, the following exit value is returned:

20049 >0 An error occurred.

20050 **CONSEQUENCES OF ERRORS**

20051 The invoking shell may terminate.

20052 **APPLICATION USAGE**

20053 There is no convenient way to enter a password into the Group Database. Use of group  
 20054 passwords is not encouraged, because by their very nature they encourage poor security  
 20055 practices. Group passwords may disappear in the future.

20056 A common implementation of *newgrp* is that the current shell uses *exec* to overlay itself with  
 20057 *newgrp*, which in turn overlays itself with a new shell after changing group. On some systems,  
 20058 however, this may not occur and *newgrp* may be invoked as a subprocess.

20059 The *newgrp* command is intended only for use from an interactive terminal. It does not offer a  
 20060 useful interface for the support of applications.

20061 The exit status of *newgrp* is generally inapplicable. If *newgrp* is used in a script, in most cases it  
 20062 will successfully invoke a new shell and the rest of the original shell script will be bypassed  
 20063 when the new shell exits. Used interactively, *newgrp* displays diagnostic messages to indicate  
 20064 problems. But usage such as:

```
20065 newgrp foo
20066 echo $?
```

20067 is not useful because the new shell might not have access to any status *newgrp* may have  
 20068 generated (and most historical systems do not provide this status). A zero status echoed here  
 20069 does not necessarily indicate that the user has changed to the new group successfully.  
 20070 Following *newgrp* with the *id* command provides a portable means of determining whether the  
 20071 group change was successful or not.

20072 **EXAMPLES**

20073 None.

20074 **FUTURE DIRECTIONS**

20075 None.

20076 **SEE ALSO**

20077 *sh*, the XSH specification description of *exec*.

20078 **CHANGE HISTORY**

20079 First released in Issue 2.

20080 **Issue 4**

20081 Aligned with the ISO/IEC 9945-2: 1993 standard.

20082 The *newgrp* utility is now mandatory; it is optional in Issue 3.



## 20083 NAME

20084        *nice* — invoke a utility with an altered system scheduling priority

## 20085 SYNOPSIS

20086        *nice* [-*n increment*] *utility* [*argument...*]

20087 OB     *nice* [-*increment*] *utility* [*argument...*]

## 20088 DESCRIPTION

20089        The *nice* utility invokes a utility, requesting that it be run with a different system scheduling  
 20090        priority (see the definition of **system scheduling priority** in the **XBD** specification, **Chapter 2,**  
 20091        **Glossary**). With no options and only if the user has appropriate privileges, the executed utility  
 20092        is run with a system scheduling priority that is some implementation-dependent quantity less  
 20093        than or equal to the system scheduling priority of the current process. If the user lacks  
 20094        appropriate privileges to affect the system scheduling priority in the requested manner, the *nice*  
 20095        utility will not affect the system scheduling priority; in this case, a warning message may be  
 20096        written to standard error, but this will not prevent the invocation of *utility* or affect the exit  
 20097        status.

## 20098 OPTIONS

20099 OB     The *nice* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines** except  
 20100        that the obsolescent version allows a multi-digit decimal integer as an option name.

20101        The following option is supported:

20102        -*n increment*

20103 OB    -*increment*

20104        Specify how the system scheduling priority of the executed utility will be adjusted. The  
 20105        *increment* option-argument is a positive or negative decimal integer that will be used to  
 20106        modify the system scheduling priority of the executed utility in an implementation-  
 20107        dependent manner.

20108        Positive *increment* values cause a lower or unchanged system scheduling priority.  
 20109        Negative *increment* values may require appropriate privileges and will cause a higher  
 20110        or unchanged system scheduling priority.

20111        The system scheduling priority is bounded in an implementation-dependent manner.  
 20112        If the requested *increment* would raise or lower the system scheduling priority of the  
 20113        executed utility beyond implementation-dependent limits, then the limit whose value  
 20114        was exceeded is used.

## 20115 OPERANDS

20116        The following operands are supported:

20117        *utility*    The name of a utility that is to be invoked. If the *utility* operand names any of the  
 20118        special built-in utilities in Section 2.14 on page 67, the results are undefined.

20119        *argument*

20120        Any string to be supplied as an argument when invoking the utility named by the  
 20121        *utility* operand.

## 20122 STDIN

20123        Not used.

## 20124 INPUT FILES

20125        None.

## 20126 ENVIRONMENT VARIABLES

20127 The following environment variables affect the execution of *nice*:

20128 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
20129 *LANG* is unset or null, the corresponding value from the implementation-dependent  
20130 default locale will be used. If any of the internationalisation variables contains an  
20131 invalid setting, the utility will behave as if none of the variables had been defined.

20132 **LC\_ALL**

20133 If set to a non-empty string value, override the values of all the other  
20134 internationalisation variables.

20135 **LC\_CTYPE**

20136 Determine the locale for the interpretation of sequences of bytes of text data as  
20137 characters (for example, single- as opposed to multi-byte characters in arguments).

20138 **LC\_MESSAGES**

20139 Determine the locale that should be used to affect the format and contents of diagnostic  
20140 messages written to standard error.

20141 EX **NLSPATH**

20142 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

20143 **PATH** Determine the search path used to locate the utility to be invoked. See the **XBD**  
20144 specification, **Chapter 6, Environment Variables**.

## 20145 ASYNCHRONOUS EVENTS

20146 Default.

20147 **STDOUT**

20148 Not used.

20149 **STDERR**

20150 Used only for diagnostic messages.

## 20151 OUTPUT FILES

20152 None.

## 20153 EXTENDED DESCRIPTION

20154 None.

## 20155 EXIT STATUS

20156 If the *utility* utility is invoked, the exit status of *nice* will be the exit status of *utility*; otherwise,  
20157 the *nice* utility will exit with one of the following values:

20158 1 - 125 An error occurred in the *nice* utility.  
20159 126 The utility specified by *utility* was found but could not be invoked.  
20160 127 The utility specified by *utility* could not be found.

## 20161 CONSEQUENCES OF ERRORS

20162 Default.

20163 **APPLICATION USAGE**

20164 Note that, in the obsolescent version, `-5` is a positive *increment*, while `--5` is a negative *increment*.

20165 The only guaranteed portable uses of this utility are:

20166 *nice utility*

20167 Run *utility* with the default lower system scheduling priority.

20168 *nice -n <positive integer> utility*

20169 Run *utility* with a lower system scheduling priority.

20170 On some systems they will have no discernible effect on the invoked utility and on some others  
20171 they will be exactly equivalent.

20172 Historical systems have frequently supported the *<positive integer>* up to 20. Since there is no  
20173 error penalty associated with guessing a number that is too high, users without access to the  
20174 system conformance document (to see what limits are actually in place) could use the historical  
20175 1 to 20 range or attempt to use very large numbers if the job should be truly low priority.

20176 The system scheduling priority value of a process can be displayed using the command:

20177 `ps -o nice`

20178 The *command*, *env*, *nice*, *nohup*, *time* and *xargs* utilities have been specified to use exit code 127 if  
20179 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
20180 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
20181 used for other meanings; most utilities use small values for “normal error conditions” and the  
20182 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
20183 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
20184 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
20185 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
20186 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
20187 any other reason.

20188 **EXAMPLES**

20189 None.

20190 **FUTURE DIRECTIONS**

20191 None.

20192 **SEE ALSO**

20193 *renice*.

20194 **CHANGE HISTORY**

20195 First released in Issue 4.

## 20196 NAME

20197 nl — line numbering filter

## 20198 SYNOPSIS

20199 EX nl [-p][-b *type*][-d *delim*][-f *type*][-h *type*][-i *incr*][-l *num*][-n *format*]  
 20200 [-s *sep*][-v *startnum*][-w *width*][*file*]

## 20201 DESCRIPTION

20202 The *nl* utility reads lines from the named *file* or the standard input if no *file* is named and  
 20203 reproduces the lines to standard output. Lines are numbered on the left. Additional  
 20204 functionality may be provided in accordance with the command options in effect.

20205 The *nl* utility views the text it reads in terms of logical pages. Line numbering is reset at the start  
 20206 of each logical page. A logical page consists of a header, a body and a footer section. Empty  
 20207 sections are valid. Different line numbering options are independently available for header,  
 20208 body and footer (for example, no numbering of header and footer lines while numbering blank  
 20209 lines only in the body).

20210 The starts of logical page sections are signalled by input lines containing nothing but the  
 20211 following delimiter characters:

| Line     | Start of |
|----------|----------|
| \: \: \: | header   |
| \: \:    | body     |
| \:       | footer   |

20217 Unless otherwise specified, *nl* assumes the text being read is in a single logical page body.

## 20218 OPTIONS

20219 OB The *nl* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 20220 that the options can be intermingled with the optional *file* operand. Only one file can be named.

20221 The following options are supported:

20222 **-b *type*** Specify which logical page body lines are to be numbered. Recognised *types* and their  
 20223 meaning are:

20224 **a** Number all lines.

20225 **t** Number only non-empty lines.

20226 **n** No line numbering.

20227 **pstring** Number only lines that contain the basic regular expression specified in *string*.

20228 The default *type* for logical page body is t (text lines numbered).

20229 **-d *delim***

20230 Specify the delimiter characters that indicate the start of a logical page section. These  
 20231 can be changed from the default characters \: to two user-specified characters. If only  
 20232 one character is entered, the second character remains the default character ":".

20233 **-f *type*** Specify the same as **b** *type* except for footer. The default for logical page footer is n (no  
 20234 lines numbered).

20235 **-h *type*** Specify the same as **b** *type* except for header. The default *type* for logical page header is  
 20236 n (no lines numbered).

20237        **-i *incr***    Specify the increment value used to number logical page lines. The default is 1.

20238        **-l *num***    Specify the number of blank lines to be considered as one. For example, **-l 2** results in  
 20239                    only the second adjacent blank line being numbered (if the appropriate **-h a**, **-b a** or  
 20240                    **-f a** option is set). The default is 1.

20241        **-n *format***  
 20242                    Specify the line numbering format. Recognised values are: **ln**, left justified, leading  
 20243                    zeros suppressed; **rn**, right justified, leading zeros suppressed; **rz**, right justified,  
 20244                    leading zeros kept. The default *format* is **rn** (right justified).

20245        **-p**        Specify that numbering should not be restarted at logical page delimiters.

20246        **-s *sep***    Specify the characters used in separating the line number and the corresponding text  
 20247                    line. The default *sep* is a tab.

20248        **-v *startnum***  
 20249                    Specify the initial value used to number logical page lines. The default is 1.

20250        **-w *width***  
 20251                    Specify the number of characters to be used for the line number. The default *width* is 6.

20252 **OPERANDS**  
 20253        The following operand is supported:

20254        ***file***        A pathname of a text file to be line-numbered.

20255 **STDIN**  
 20256        The standard input is a text file that is used if no *file* operand is given.

20257 **INPUT FILES**  
 20258        The input file named by the *file* operand is a text file.

20259 **ENVIRONMENT VARIABLES**  
 20260        The following environment variables affect the execution of *nl*:

20261        **LANG**        Provide a default value for the internationalisation variables that are unset or null. If  
 20262                    **LANG** is unset or null, the corresponding value from the implementation-dependent  
 20263                    default locale will be used. If any of the internationalisation variables contains an  
 20264                    invalid setting, the utility will behave as if none of the variables had been defined.

20265        **LC\_ALL**  
 20266                    If set to a non-empty string value, override the values of all the other  
 20267                    internationalisation variables.

20268        **LC\_COLLATE**  
 20269                    Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 20270                    character collating elements within regular expressions.

20271        **LC\_CTYPE**  
 20272                    Determine the locale for the interpretation of sequences of bytes of text data as  
 20273                    characters (for example, single- as opposed to multi-byte characters in arguments and  
 20274                    input files), the behaviour of character classes within regular expressions, and for  
 20275                    deciding which characters are in character class **graph** (for the **-b t**, **-f t** and **-h t**  
 20276                    options).

20277        **LC\_MESSAGES**  
 20278                    Determine the locale that should be used to affect the format and contents of diagnostic  
 20279                    messages written to standard error.

20280 **NLSPATH**  
 20281 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

20282 **ASYNCHRONOUS EVENTS**  
 20283 Default.

20284 **STDOUT**  
 20285 The standard output is a text file in the following format:  
 20286 "%s%s%s", <line number>, <separator>, <input line>  
 20287 where <line number> is one of the following numeric formats:  
 20288 %6d when the **rn** format is used (the default; see **-n**).  
 20289 %06d when the **rz** format is used.  
 20290 %-6d when the **ln** format is used.  
 20291 <empty>  
 20292 when line numbers are suppressed for a portion of the page; the <separator> is also  
 20293 suppressed.  
 20294 In the preceding list, the number 6 is the default width; the **-w** option can change this value.

20295 **STDERR**  
 20296 Used only for diagnostic messages.

20297 **OUTPUT FILES**  
 20298 None.

20299 **EXTENDED DESCRIPTION**  
 20300 None.

20301 **EXIT STATUS**  
 20302 The following exit values are returned:  
 20303 0 Successful completion.  
 20304 >0 An error occurred.

20305 **CONSEQUENCES OF ERRORS**  
 20306 Default.

20307 **APPLICATION USAGE**  
 20308 In using the **-d delim** option, care should be taken to escape characters that have special  
 20309 meaning to the command interpreter.

20310 **EXAMPLES**  
 20311 The command:  
 20312 nl -v 10 -i 10 -d \!+ file1  
 20313 will number *file1* starting at line number 10 with an increment of 10. The logical page delimiter  
 20314 is "!+". Note that the "!" has to be escaped when using *cs*h as a command interpreter because of  
 20315 its history substitution syntax. For *ksh* and *sh* the escape is not necessary, but will not do any  
 20316 harm.

20317 **FUTURE DIRECTIONS**  
 20318 The intermingling of the *file* operand with the options is an obsolescent feature that will be  
 20319 removed from a future issue.

20320 **SEE ALSO**

20321 *pr.*

20322 **CHANGE HISTORY**

20323 First released in Issue 2.

20324 **Issue 4**

20325 Format reorganised.

20326 Utility Syntax Guideline support mandated.

20327 Internationalised environment variable support mandated.

20328 **Issue 5**

20329 The option [-f type] is added to the SYNOPSIS. The option descriptions are presented in  
20330 alphabetic order. The description of **-bt** is changed to “Number only non-empty lines”.

## 20331 NAME

20332 nm — write the name list of an object file (DEVELOPMENT)

## 20333 SYNOPSIS

20334 EX nm [-APv][-efox][-g|-u][-t *format*] *file*...

## 20335 DESCRIPTION

20336 The *nm* utility displays symbolic information appearing in the object file, executable file or  
 20337 object-file library named by *file*. If no symbolic information is available for a valid input file, the  
 20338 *nm* utility will report that fact, but not consider it an error condition.

20339 EX The default base used when numeric values are written is decimal.

## 20340 OPTIONS

20341 The *nm* utility supports the XBD specification, Section 10.2, Utility Syntax Guidelines.

20342 The following options are supported:

20343 -A Write the full pathname or library name of an object on each line.

20344 EX -e Write only external (global) and static symbol information.

20345 EX -f Produce full output. Write redundant symbols (*.text*, *.data* and *.bss*), normally  
 20346 suppressed.

20347 -g Write only external (global) symbol information.

20348 EX -o Write numeric values in octal (equivalent to -t o).

20349 -P Write information in a portable output format, as specified in the STDOUT section.

20350 -t *format*

20351 Write each numeric value in the specified format. The format is dependent on the  
 20352 single character used as the *format* option-argument:

20353 EX d The offset is written in decimal (default).

20354 o The offset is written in octal.

20355 x The offset is written in hexadecimal.

20356 -u Write only undefined symbols.

20357 -v Sort output by value instead of alphabetically.

20358 EX -x Write numeric values in hexadecimal (equivalent to -t x).

## 20359 OPERANDS

20360 The following operand is supported:

20361 *file* A pathname of an object file, executable file or object-file library.

## 20362 STDIN

20363 See the INPUT FILES section.

## 20364 INPUT FILES

20365 The input file must be an object file, an object-file library whose format is the same as those  
 20366 produced by the *ar* utility for link editing, or an executable file. The *nm* utility may accept  
 20367 additional implementation-dependent object library formats for the input file.



20368 **ENVIRONMENT VARIABLES**

20369 The following environment variables affect the execution of *nm*:

20370 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 20371 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 20372 default locale will be used. If any of the internationalisation variables contains an  
 20373 invalid setting, the utility will behave as if none of the variables had been defined.

20374 **LC\_ALL**

20375 If set to a non-empty string value, override the values of all the other  
 20376 internationalisation variables.

20377 **LC\_COLLATE**

20378 Determine the locale for character collation information for the symbol-name and  
 20379 symbol-value collation sequences.

20380 **LC\_CTYPE**

20381 Determine the locale for the interpretation of sequences of bytes of text data as  
 20382 characters (for example, single- as opposed to multi-byte characters in arguments).

20383 **LC\_MESSAGES**

20384 Determine the locale that should be used to affect the format and contents of diagnostic  
 20385 messages written to standard error.

20386 EX **NLSPATH**

20387 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

20388 **ASYNCHRONOUS EVENTS**

20389 Default.

20390 **STDOUT**

20391 If symbolic information is present in the input files, then for each file or for each member of an  
 20392 archive, the *nm* utility will write the following information to standard output. By default, the  
 20393 format is unspecified, but the output is sorted alphabetically by symbol name.

- 20394 • Library or object name, if **-A** is specified.
- 20395 • Symbol name.
- 20396 • Symbol type, which is either one of the following single characters or an implementation-  
 20397 dependent type represented by a single character:
- 20398 **A** Global absolute symbol.
- 20399 **a** Local absolute symbol.
- 20400 **B** Global “bss” (that is, uninitialised data space) symbol.
- 20401 **b** Local bss symbol.
- 20402 **D** Global data symbol.
- 20403 **d** Local data symbol.
- 20404 **T** Global text symbol.
- 20405 **t** Local text symbol.
- 20406 **U** Undefined symbol.
- 20407 • Value of the symbol.

20408       • The size associated with the symbol, if applicable.

20409       This information may be supplemented by additional information specific to the  
20410       implementation.

20411       If the **-P** option is specified, the previous information is displayed using the following portable  
20412       format. The three versions differ depending on whether **-t d**, **-t o** or **-t x** was specified,  
20413       respectively:

20414            "%s%s %s %d %d\n", <library/object name>, <name>, <type>,  
20415            <value>, <size>

20416            "%s%s %s %o %o\n", <library/object name>, <name>, <type>,  
20417            <value>, <size>

20418            "%s%s %s %x %x\n", <library/object name>, <name>, <type>,  
20419            <value>, <size>

20420       where <library/object name> is formatted as follows:

20421       • If **-A** is not specified, <library/object name> is an empty string.

20422       • If **-A** is specified and the corresponding *file* operand does not name a library:

20423            "%s: ", <file>

20424       • If **-A** is specified and the corresponding *file* operand names a library. In this case,  
20425       <object file> names the object file in the library containing the symbol being described:

20426            "%s[%s]: ", <file>, <object file>

20427       If **-A** is not specified, then if more than one *file* operand is specified or if only one *file* operand is  
20428       specified and it names a library, *nm* will write a line identifying the object containing the  
20429       following symbols before the lines containing those symbols, in the form:

20430       • If the corresponding *file* operand does not name a library:

20431            "%s:\n", <file>

20432       • If the corresponding *file* operand names a library; in this case, <object file> is the name of the  
20433       file in the library containing the following symbols:

20434            "%s[%s]:\n", <file>, <object file>

20435       If **-P** is specified, but **-t** is not, the format is as if **-t x** had been specified.

20436       **STDERR**

20437       Used only for diagnostic messages.

20438       **OUTPUT FILES**

20439       None.

20440       **EXTENDED DESCRIPTION**

20441       None.

20442       **EXIT STATUS**

20443       The following exit values are returned:

20444            0   Successful completion.

20445            >0   An error occurred.

20446       **CONSEQUENCES OF ERRORS**

20447       Default.

20448 **APPLICATION USAGE**

20449           Mechanisms for dynamic linking make this utility less meaningful when applied to an  
20450           executable file because a dynamically linked executable may omit numerous library routines  
20451           that would be found in a statically linked executable.

20452 **EXAMPLES**

20453           None.

20454 **FUTURE DIRECTIONS**

20455           None.

20456 **SEE ALSO**

20457           *ar*, *c89*.

20458 **CHANGE HISTORY**

20459           First released in Issue 2.

20460 **Issue 4**

20461           Aligned with the ISO/IEC 9945-2: 1993 standard.

## 20462 NAME

20463       nohup — invoke a utility immune to hangups

## 20464 SYNOPSIS

20465       nohup *utility* [*argument...*]

## 20466 DESCRIPTION

20467       The *nohup* utility will invoke the utility named by the *utility* operand with arguments supplied  
 20468 as the *argument* operands. At the time the named *utility* is invoked, the SIGHUP signal is set to  
 20469 be ignored.

20470       If the standard output is a terminal, all output written by the named *utility* to its standard output  
 20471 will be appended to the end of the file **nohup.out** in the current directory. If **nohup.out** cannot  
 20472 be created or opened for appending, the output will be appended to the end of the file  
 20473 **nohup.out** in the directory specified by the *HOME* environment variable. If neither file can be  
 20474 created or opened for appending, *utility* will not be invoked. If a file is created, the file's  
 20475 permission bits will be set to S\_IRUSR | S\_IWUSR.

20476       If the standard error is a terminal, all output written by the named *utility* to its standard error  
 20477 will be redirected to the same file descriptor as the standard output.

## 20478 OPTIONS

20479       None.

## 20480 OPERANDS

20481       The following operands are supported:

20482       *utility*   The name of a utility that is to be invoked. If the *utility* operand names any of the  
 20483 special built-in utilities in Section 2.14 on page 67, the results are undefined.

20484       *argument*   Any string to be supplied as an argument when invoking the utility named by the  
 20485 *utility* operand.  
 20486

## 20487 STDIN

20488       Not used.

## 20489 INPUT FILES

20490       None.

## 20491 ENVIRONMENT VARIABLES

20492       The following environment variables affect the execution of *nohup*:

20493       *HOME*   Determine the pathname of the user's home directory: if the output file **nohup.out**  
 20494 cannot be created in the current directory, the *nohup* utility will use the directory  
 20495 named by *HOME* to create the file.

20496       *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 20497 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 20498 default locale will be used. If any of the internationalisation variables contains an  
 20499 invalid setting, the utility will behave as if none of the variables had been defined.

20500       *LC\_ALL*

20501       If set to a non-empty string value, override the values of all the other  
 20502 internationalisation variables.

20503       *LC\_CTYPE*

20504       Determine the locale for the interpretation of sequences of bytes of text data as  
 20505 characters (for example, single- as opposed to multi-byte characters in arguments).

20506 **LC\_MESSAGES**  
 20507 Determine the locale that should be used to affect the format and contents of diagnostic  
 20508 messages written to standard error.

20509 EX **NLSPATH**  
 20510 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

20511 **PATH** Determine the search path that will be used to locate the utility to be invoked. See the  
 20512 XBD specification, **Chapter 6, Environment Variables**.

20513 **ASYNCHRONOUS EVENTS**  
 20514 The *nohup* utility will take the standard action for all signals except that SIGHUP will be ignored.

20515 **STDOUT**  
 20516 If the standard output is not a terminal, the standard output of *nohup* will be the standard output  
 20517 generated by the execution of the *utility* specified by the operands. Otherwise, nothing will be  
 20518 written to the standard output.

20519 **STDERR**  
 20520 If the standard output is a terminal, a message will be written to the standard error, indicating  
 20521 the name of the file to which the output is being appended. The name of the file will be either  
 20522 **nohup.out** or **\$HOME/nohup.out**.

20523 **OUTPUT FILES**  
 20524 If the standard output is a terminal, all output written by the named *utility* to the standard  
 20525 output and standard error is appended to the file **nohup.out**, which is created if it does not  
 20526 already exist.

20527 **EXTENDED DESCRIPTION**  
 20528 None.

20529 **EXIT STATUS**  
 20530 The following exit values are returned:

20531 126 The utility specified by *utility* was found but could not be invoked.  
 20532 127 An error occurred in the *nohup* utility or the utility specified by *utility* could not be  
 20533 found.

20534 Otherwise, the exit status of *nohup* will be that of the utility specified by the *utility* operand.

20535 **CONSEQUENCES OF ERRORS**  
 20536 Default.

20537 **APPLICATION USAGE**  
 20538 The *command*, *env*, *nice*, *nohup*, *time* and *xargs* utilities have been specified to use exit code 127 if  
 20539 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
 20540 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
 20541 used for other meanings; most utilities use small values for “normal error conditions” and the  
 20542 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 20543 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
 20544 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 20545 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 20546 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
 20547 any other reason.

20548 **EXAMPLES**

20549 It is frequently desirable to apply *nohup* to pipelines or lists of commands. This can be done by  
20550 placing pipelines and command lists in a single file; this file can then be invoked as a utility, and  
20551 the *nohup* applies to everything in the file.

20552 Alternatively, the following command can be used to apply *nohup* to a complex command:

20553 `nohup sh -c 'complex-command-line'`

20554 **FUTURE DIRECTIONS**

20555 None.

20556 **SEE ALSO**

20557 *sh*, the **XSH** specification description of *signal()*.

20558 **CHANGE HISTORY**

20559 First released in Issue 2.

20560 **Issue 4**

20561 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 20562 NAME

20563 od — dump files in various formats

## 20564 SYNOPSIS

20565 od [-v][-A *address\_base*][-j *skip*][-N *count*][-t *type\_string*]  
 20566 [*file*...]

20567 EX od [-bcdosx][*file*] [[+]*offset*][.][b]]

## 20568 DESCRIPTION

20569 The *od* utility will write the contents of its input files to standard output in a user-specified  
 20570 format.

## 20571 OPTIONS

20572 The *od* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 20573 EX that the order of presentation of the **-t** options and the **-bcdosx** options is significant.

20574 The following options are supported:

20575 **-A *address\_base***

20576 Specify the input offset base (see the EXTENDED DESCRIPTION section). The  
 20577 *address\_base* option-argument must be a character. The characters d, o and x specify  
 20578 that the offset base will be written in decimal, octal or hexadecimal, respectively. The  
 20579 character n specifies that the offset will not be written.

20580 EX **-b** Interpret bytes in octal. This is equivalent to **-t o1**.

20581 EX **-c** Interpret bytes as characters specified by the current setting of the LC\_CTYPE category.  
 20582 Certain non-graphic characters appear as C escapes: NUL=**\0**, BS=**\b**, FF=**\f**, NL=**\n**,  
 20583 CR=**\r**, HT=**\t**; others appear as 3-digit octal numbers.

20584 EX **-d** Interpret *words* (two-byte units) in unsigned decimal. This is equivalent to **-t u2**.

20585 **-j *skip*** Jump over *skip* bytes from the beginning of the input. The *od* utility will read or seek  
 20586 past the first *skip* bytes in the concatenated input files. If the combined input is not at  
 20587 least *skip* bytes long, the *od* utility will write a diagnostic message to standard error and  
 20588 exit with a non-zero exit status.

20589 By default, the *skip* option-argument is interpreted as a decimal number. With a  
 20590 leading 0x or 0X, the offset is interpreted as a hexadecimal number; otherwise, with a  
 20591 leading 0, the offset will be interpreted as an octal number. Appending the character b,  
 20592 k or m to offset will cause it to be interpreted as a multiple of 512, 1024 or 1048576  
 20593 EX bytes, respectively. If the *skip* number is hexadecimal, any appended b is considered to  
 20594 be the final hexadecimal digit.

20595 **-N *count***

20596 Format no more than *count* bytes of input. By default, *count* is interpreted as a decimal  
 20597 number. With a leading 0x or 0X, *count* is interpreted as a hexadecimal number;  
 20598 otherwise, with a leading 0, it is interpreted as an octal number. If *count* bytes of input  
 20599 (after successfully skipping, if **-j *skip*** is specified) are not available, it will not be  
 20600 considered an error; the *od* utility will format the input that is available.

20601 EX **-o** Interpret *words* (two-byte units) in octal. This is equivalent to **-t o2**.

20602 EX PI **-s** Interpret *words* (two-byte units) in signed decimal. This is equivalent to  
 20603 **-t d2**.

20604        **-t** *type\_string*  
 20605            Specify one or more output types (see the EXTENDED DESCRIPTION section). The  
 20606            *type\_string* option-argument must be a string specifying the types to be used when  
 20607            writing the input data. The string must consist of the type specification characters a, c,  
 20608            d, f, o, u and x, specifying named character, character, signed decimal, floating point,  
 20609            octal, unsigned decimal and hexadecimal, respectively. The type specification  
 20610            characters d, f, o, u and x can be followed by an optional unsigned decimal integer that  
 20611            specifies the number of bytes to be transformed by each instance of the output type.  
 20612            The type specification character f can be followed by an optional F, D or L indicating  
 20613            that the conversion should be applied to an item of type *float*, *double* or *long double*,  
 20614            respectively. The type specification characters d, o, u and x can be followed by an  
 20615            optional C, S, I or L indicating that the conversion should be applied to an item of type  
 20616            **char**, **short**, **int** or **long**, respectively. Multiple types can be concatenated within the  
 20617            same *type\_string* and multiple **-t** options can be specified. Output lines are written for  
 20618            each type specified in the order in which the type specification characters are specified.

20619        **-v**        Write all input data. Without the **-v** option, any number of groups of output lines,  
 20620            which would be identical to the immediately preceding group of output lines (except  
 20621            for the byte offsets), will be replaced with a line containing only an asterisk (\*).

20622 EX        **-x**        Interpret *words* (two-byte units) in hexadecimal. This is equivalent to **-t x2**.

20623 EX        Multiple types can be specified by using multiple **-bcdostx** options. Output lines are written for  
 20624            each type specified in the order in which the types are specified.

20625 **OPERANDS**  
 20626        The following operands are supported:

20627        *file*        A pathname of a file to be written. If no file operands are specified, the standard input  
 20628            will be used. If the first character of *file* is a plus sign (+) or the first character of the first  
 20629            file operand is numeric, no more than two operands are given, and none of the **-A**, **-j**,  
 20630 EX            **-N** or **-t** options is specified, the operand is assumed to be an *offset*.

20631 EX        **[+]*offset*[.][**b**]**  
 20632            The *offset* operand specifies the offset in the file where dumping is to commence. This  
 20633            operand is normally interpreted as octal bytes. If **.** is appended, the offset is interpreted  
 20634            in decimal. If **b** is appended, the offset is interpreted in units of 512 bytes. If the *file*  
 20635            argument is omitted, and none of the **-A**, **-j**, **-N** or **-t** options is specified, the offset  
 20636            argument must be preceded by **+**.

20637 **STDIN**  
 20638        The standard input is used only if no *file* operands are specified. See the INPUT FILES section.

20639 **INPUT FILES**  
 20640        The input files can be any file type.

20641 **ENVIRONMENT VARIABLES**  
 20642        The following environment variables affect the execution of *od*:

20643        *LANG*        Provide a default value for the internationalisation variables that are unset or null. If  
 20644            *LANG* is unset or null, the corresponding value from the implementation-dependent  
 20645            default locale will be used. If any of the internationalisation variables contains an  
 20646            invalid setting, the utility will behave as if none of the variables had been defined.

20647        *LC\_ALL*  
 20648            If set to a non-empty string value, override the values of all the other  
 20649            internationalisation variables.



20650 **LC\_CTYPE**

20651 Determine the locale for the interpretation of sequences of bytes of text data as  
 20652 characters (for example, single- as opposed to multi-byte characters in arguments and  
 20653 input files).

20654 **LC\_MESSAGES**

20655 Determine the locale that should be used to affect the format and contents of diagnostic  
 20656 messages written to standard error.

20657 **LC\_NUMERIC**

20658 Determine the locale for selecting the radix character used when writing floating-point  
 20659 formatted output.

20660 EX **NLSPATH**

20661 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

20662 **ASYNCHRONOUS EVENTS**

20663 Default.

20664 **STDOUT**

20665 See the EXTENDED DESCRIPTION section.

20666 **STDERR**

20667 Used only for diagnostic messages.

20668 **OUTPUT FILES**

20669 None.

20670 **EXTENDED DESCRIPTION**

20671 The *od* utility copies sequentially each input file to standard output, transforming the input data  
 20672 EX according to the output types specified by the **-t** options or the **-bcdosx** options. If no output  
 20673 type is specified, the default output is as if **-t o2** had been specified.

20674 The number of bytes transformed by the output type specifier *c* may be variable depending on  
 20675 the **LC\_CTYPE** category.

20676 The default number of bytes transformed by output type specifiers *d*, *f*, *o*, *u* and *x* will  
 20677 correspond to the various C-language types as follows. If the *c89* compiler is present on the  
 20678 system, these specifiers will correspond to the sizes used by default in that compiler. Otherwise,  
 20679 these sizes are implementation-dependent.

- 20680 • For the type specifier characters *d*, *o*, *u* and *x*, the default number of bytes will correspond to
- 20681 the size of the underlying implementation's basic integral data type. For these specifier
- 20682 characters, the implementation will support values of the optional number of bytes to be
- 20683 converted corresponding to the number of bytes in the C-language types **char**, **short**, **int** and
- 20684 **long**. These numbers can also be specified by an application as the characters *C*, *S*, *I* and *L*,
- 20685 respectively. The byte order used when interpreting numeric values is implementation-
- 20686 dependent, but will correspond to the order in which a constant of the corresponding type is
- 20687 stored in memory on the system.

- 20688 • For the type specifier character *f*, the default number of bytes will correspond to the number
- 20689 of bytes in the underlying implementation's basic double precision floating-point data type.
- 20690 The implementation will support values of the optional number of bytes to be converted
- 20691 corresponding to the number of bytes in the C-language types **float**, **double** and **long**
- 20692 **double**. These numbers can also be specified by an application as the characters *F*, *D* and *L*,
- 20693 respectively.

20694 The type specifier character *a* specifies that bytes are interpreted as named characters from the  
 20695 International Reference Version (IRV) of the ISO/IEC 646:1991 standard. Only the least

significant seven bits of each byte will be used for this type specification. Bytes with the values listed in the following table will be written using the corresponding names for those characters.

| Value | Name | Value | Name | Value | Name      | Value | Name |
|-------|------|-------|------|-------|-----------|-------|------|
| \000  | nul  | \001  | soh  | \002  | stx       | \003  | etx  |
| \004  | eot  | \005  | enq  | \006  | ack       | \007  | bel  |
| \010  | bs   | \011  | ht   | \012  | lf or nl* | \013  | vt   |
| \014  | ff   | \015  | cr   | \016  | so        | \017  | si   |
| \020  | dle  | \021  | dc1  | \022  | dc2       | \023  | dc3  |
| \024  | dc4  | \025  | nak  | \026  | syn       | \027  | etb  |
| \030  | can  | \031  | em   | \032  | sub       | \033  | esc  |
| \034  | fs   | \035  | gs   | \036  | rs        | \037  | us   |
| \040  | sp   | \177  | del  |       |           |       |      |

**Table 3-10** Named Characters in *od*

**Note:** The \012 value may be written either as lf or nl.

The type specifier character *c* specifies that bytes will be interpreted as characters specified by the current setting of the LC\_CTYPE locale category. Characters listed in the table in the **XBD** specification, **Chapter 3, File Format Notation** (\, \a, \b, \f, \n, \r, \t, \v) will be written as the corresponding escape sequences, except that backslash will be written as a single backslash and a NUL will be written as \0. Other non-printable characters will be written as one three-digit octal number for each byte in the character. If the size of a byte on the system is greater than nine bits, the format used for non-printable characters is implementation-dependent. Printable multi-byte characters will be written in the area corresponding to the first byte of the character; the two-character sequence **\*\*** will be written in the area corresponding to each remaining byte in the character, as an indication that the character is continued. When either the **-j skip** or **-N count** option is specified along with the *c* type specifier, and this results in an attempt to start or finish in the middle of a multi-byte character, the result is implementation-dependent.

The input data is manipulated in blocks, where a block is defined as a multiple of the least common multiple of the number of bytes transformed by the specified output types. If the least common multiple is greater than 16, the results are unspecified. Each input block will be written as transformed by each output type, one per written line, in the order that the output types were specified. If the input block size is larger than the number of bytes transformed by the output type, the output type will sequentially transform the parts of the input block, and the output from each of the transformations will be separated by one or more blank characters.

If, as a result of the specification of the **-N** option or end-of-file being reached on the last input file, input data only partially satisfies an output type, the input will be extended sufficiently with null bytes to write the last byte of the input.

Unless **-A n** is specified, the first output line produced for each input block will be preceded by the input offset, cumulative across input files, of the next byte to be written. The format of the input offset is unspecified; however, it will not contain any blank characters, will start at the first character of the output line, and will be followed by one or more blank characters. In addition, the offset of the byte following the last byte written will be written after all the input data has been processed, but will not be followed by any blank characters.

If no **-A** option is specified, the input offset base is unspecified.

**20741 EXIT STATUS**

20742 The following exit values are returned:

20743 0 All input files were processed successfully.

20744 >0 An error occurred.

**20745 CONSEQUENCES OF ERRORS**

20746 Default.

**20747 APPLICATION USAGE**

20748 Applications are warned not to use filenames starting with + or a first operand starting with a  
 20749 numeric character so that the old functionality can be maintained by implementations, unless  
 20750 they specify one of the new options specified by the ISO/IEC 9945-2:1993 standard. To  
 20751 guarantee that one of these filenames will always be interpreted as a filename, an application  
 20752 could always specify the address base format with the -A option.

**20753 EXAMPLES**

20754 If a file containing 128 bytes with decimal values zero to 127, in increasing order, is supplied as  
 20755 standard input to the command:

20756 `od -A d -t a`

20757 on an implementation using an input block size of 16 bytes, the standard output, independent of  
 20758 the current locale setting, would be similar to:

```

20759 00000000 nul soh stx etx eot enq ack bel bs ht nl vt ff cr so si
20760 0000016 dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
20761 0000032 sp ! " # $ % & ' () * + , - . /
20762 0000048 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
20763 0000064 @ A B C D E F G H I J K L M N O
20764 0000080 P Q R S T U V W X Y Z [\] ^ _
20765 0000096 ` a b c d e f g h i j k l m n o
20766 0000112 p q r s t u v w x y z { | } ~ del
20767 0000128

```

20768 Note that this specification allows **nl** or **If** to be used as the name for the ISO/IEC 646:1991  
 20769 standard IRV character with decimal value 10. The IRV names this character **If** (line feed), but  
 20770 traditional implementations have referred to this character as newline (nl) and the POSIX locale  
 20771 character set symbolic name for the corresponding character is a newline character.

20772 The command:

20773 `od -A o -t o2x2x -n 18`

20774 on a system with 32-bit words and an implementation using an input block size of 16 bytes  
 20775 could write 18 bytes in approximately the following format:

```

20776 0000000 032056 031440 041123 042040 052516 044530 020043 031464
20777 342e 3320 4253 4420 554e 4958 2023 3334
20778 342e3320 42534420 554e4958 20233334
20779 0000020 032472
20780 353a
20781 353a0000
20782 0000022

```

20783 The command:

20784 `od -A d -t f -t o4 -t x4 -n 24 -j 0x15`

20785 on a system with 64-bit doubles (for example, the IEEE Std. 754 double precision floating-point  
20786 format) would skip 21 bytes of input data and then write 24 bytes in approximately the  
20787 following format:

```
20788 00000000 1.0000000000000000e+00 1.5735000000000000e+01
20789 07774000000 000000000000 10013674121 35341217270
20790 3ff00000 00000000 402f3851 eb851eb8
20791 0000016 1.4066823000000000e+02
20792 10030312542 04370303230
20793 40619562 23e18698
20794 0000024
```

#### 20795 **FUTURE DIRECTIONS**

20796 All option and operand interfaces marked as extensions may be withdrawn in a future issue.

20797 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
20798 interface definition to the IEEE PASC 1003.2 Interpretations Committee which is identifying the  
20799 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
20800 finalised.

#### 20801 **SEE ALSO**

20802 *sed*.

#### 20803 **CHANGE HISTORY**

20804 First released in Issue 2.

#### 20805 **Issue 4**

20806 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 20807 **Issue 4, Version 2**

20808 The description of the `-c` option is made dependent on the current setting of the LC\_CTYPE  
20809 category, and a reference to the POSIX locale is deleted.

#### 20810 **Issue 5**

20811 In the description of the `-c` option, the phrase “This is equivalent to `-t c`.” is deleted.

20812 The FUTURE DIRECTIONS section has been modified.

20813 **NAME**20814           pack — compress files (**LEGACY**)20815 **SYNOPSIS**20816 EX       pack [-f][-] *file...*20817 **DESCRIPTION**

20818       The *pack* utility attempts to store the specified files in a compressed form. Each input file is  
 20819       replaced by a packed file *file.z*. If the invoking process has appropriate privileges, the  
 20820       ownership, modes, access time, and modification time of the original file are preserved. If *pack* is  
 20821       successful, *file* will be removed. Packed files can be restored to their original form using *unpack*  
 20822       or *pcat*.

20823       No packing will occur if:

- 20824           • The file appears to be already packed.
- 20825           • The filename has more than {NAME\_MAX}-2 bytes.
- 20826           • The file has links.
- 20827           • The file is a directory.
- 20828           • The file cannot be opened.
- 20829           • The file is empty.
- 20830           • No disk storage will be saved by packing.
- 20831           • A file called *file.z* already exists.
- 20832           • The *.z* file cannot be created.
- 20833           • An I/O error occurred during processing.

20834       The last segment of the filename must contain no more than {NAME\_MAX}-2 bytes to allow  
 20835       space for the appended *.z* extension.

20836 **OPTIONS**

20837       The *pack* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 20838       following option is supported:

20839       -f       Force packing of files. This is useful for causing an entire directory to be packed even if  
 20840       some of the files will not benefit.

20841 **OPERANDS**

20842       The following operands are supported:

20843       -       Sets an internal flag that causes the number of times each byte is used, its relative  
 20844       frequency and the code for the byte to be written to standard output. Additional  
 20845       occurrences of - in place of *file* will cause the internal flag to be set and reset.

20846       *file*   A pathname of a file to be packed; *file* can include or omit the *.z* suffix.

20847 **STDIN**

20848       Not used.

20849 **INPUT FILES**

20850       The input files are regular files.

20851 **ENVIRONMENT VARIABLES**

20852 The following environment variables may affect the execution of *pack*:

20853 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 20854 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 20855 default locale will be used. If any of the internationalisation variables contains an  
 20856 invalid setting, the utility will behave as if none of the variables had been defined.

20857 *LC\_ALL*

20858 If set to a non-empty string value, override the values of all the other  
 20859 internationalisation variables.

20860 *LC\_CTYPE*

20861 Determine the locale for the interpretation of sequences of bytes of text data as  
 20862 characters (for example, single- as opposed to multi-byte characters in arguments).

20863 *LC\_MESSAGES*

20864 Determine the locale that should be used to affect the format and contents of diagnostic  
 20865 messages written to standard error, and informative messages written to standard  
 20866 output.

20867 *NLSPATH*

20868 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

20869 **ASYNCHRONOUS EVENTS**

20870 If an error or signal occurs, the *file.z* file is not created and the original file is unchanged.

20871 **STDOUT**

20872 The standard output is a text file containing one line for each file packed, with the following  
 20873 format used in the POSIX locale:

20874 "pack: %s: %2.1f%% Compression\n", *file*, <percent compression>

20875 or:

20876 "pack: %s: no saving - file unchanged\n", *file*

20877 If the *-* operand is specified and the internal flag is set, additional messages of unspecified  
 20878 format will be written to standard output as indicated in the OPERANDS section.

20879 **STDERR**

20880 Used only for diagnostic messages.

20881 **OUTPUT FILES**

20882 Packed files of unspecified format are created with names of the form *file.z*.

20883 **EXTENDED DESCRIPTION**

20884 None.

20885 **EXIT STATUS**

20886 The following exit values are returned:

20887 0 Successful completion.

20888 >0 An error occurred.

20889 **CONSEQUENCES OF ERRORS**

20890 Default.

20891 **APPLICATION USAGE**

20892 The amount of compression obtained depends on the size of the input file and the character  
 20893 frequency distribution. Because a decoding tree may form the first part of each *.z* file, it is

|       |                                                                                                               |  |
|-------|---------------------------------------------------------------------------------------------------------------|--|
| 20894 | usually not worthwhile to pack small files, unless the character frequency distribution is very               |  |
| 20895 | skewed, which may occur with printer plots or pictures.                                                       |  |
| 20896 | Typically, text files are reduced to 60–75% of their original size. Object files, which use a larger          |  |
| 20897 | character set and have a more uniform distribution of characters, show little compression, the                |  |
| 20898 | packed versions typically being about 90% of the original size.                                               |  |
| 20899 | Packed files are not necessarily portable to other systems.                                                   |  |
| 20900 | Applications should migrate to the <i>compress</i> utility. The <i>compress</i> utility offers two advantages |  |
| 20901 | over <i>pack</i> :                                                                                            |  |
| 20902 | • The algorithm used to create the output files is frequently more effective in reducing the                  |  |
| 20903 | sizes of files.                                                                                               |  |
| 20904 | • The <i>compress</i> utility can compress data from its standard input, not just a named regular file.       |  |
| 20905 | Thus, it is useful in pipelines.                                                                              |  |
| 20906 | <b>EXAMPLES</b>                                                                                               |  |
| 20907 | None.                                                                                                         |  |
| 20908 | <b>FUTURE DIRECTIONS</b>                                                                                      |  |
| 20909 | None.                                                                                                         |  |
| 20910 | <b>SEE ALSO</b>                                                                                               |  |
| 20911 | <i>pcat, unpack, compress.</i>                                                                                |  |
| 20912 | <b>CHANGE HISTORY</b>                                                                                         |  |
| 20913 | First released in Issue 2.                                                                                    |  |
| 20914 | <b>Issue 4</b>                                                                                                |  |
| 20915 | Format reorganised.                                                                                           |  |
| 20916 | Split into a separate description.                                                                            |  |
| 20917 | Marked TO BE WITHDRAWN.                                                                                       |  |
| 20918 | Utility Syntax Guideline support mandated.                                                                    |  |
| 20919 | Internationalised environment variable support made optional.                                                 |  |
| 20920 | <b>Issue 4, Version 2</b>                                                                                     |  |
| 20921 | The DESCRIPTION section is clarified to state that the ownership, modes, access time, and                     |  |
| 20922 | modification time of the original file are preserved if the invoking process has appropriate                  |  |
| 20923 | privileges.                                                                                                   |  |
| 20924 | <b>Issue 5</b>                                                                                                |  |
| 20925 | Marked LEGACY.                                                                                                |  |

## 20926 NAME

20927 paste — merge corresponding or subsequent lines of files

## 20928 SYNOPSIS

20929 paste [-s][-d *list*] *file...*

## 20930 DESCRIPTION

20931 The *paste* utility will concatenate the corresponding lines of the given input files, and write the  
 20932 resulting lines to standard output.

20933 The default operation of *paste* will concatenate the corresponding lines of the input files. The  
 20934 newline character of every line except the line from the last input file will be replaced with a tab  
 20935 character.

20936 If an end-of-file condition is detected on one or more input files, but not all input files, *paste* will  
 20937 behave as though empty lines were read from the files on which end-of-file was detected, unless  
 20938 the **-s** option is specified.

## 20939 OPTIONS

20940 The *paste* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

20941 The following options are supported:

20942 **-d *list*** Unless a backslash character appears in *list*, each character in *list* is an element  
 20943 specifying a delimiter character. If a backslash character appears in *list*, the backslash  
 20944 character and one or more characters following it are an element specifying a delimiter  
 20945 character as described below. These elements specify one or more delimiters to use,  
 20946 instead of the default tab character, to replace the newline character of the input lines.  
 20947 The elements in *list* are used circularly; that is, when the list is exhausted the first  
 20948 element from the list is reused. When the **-s** option is specified:

- 20949 • The last newline character in a file will not be modified.
- 20950 • The delimiter will be reset to the first element of *list* after each *file* operand is
- 20951 processed.

20952 When the **-s** option is not specified:

- 20953 • The newline characters in the file specified by the last *file* operand will not be
- 20954 modified.
- 20955 • The delimiter will be reset to the first element of *list* each time a line is processed
- 20956 from each file.

20957 If a backslash character appears in *list*, it and the character following it will be used to  
 20958 represent the following delimiter characters:

20959 **\n** Newline character.

20960 **\t** Tab character.

20961 **\\** Backslash character.

20962 **\0** Empty string (not a null character). If **\0** is immediately followed by the  
 20963 character *x*, the character *X*, or any character defined by the **LC\_CTYPE digit**  
 20964 keyword (see the **XBD** specification, **Chapter 5, Locale**), the results are  
 20965 unspecified.

20966 If any other characters follow the backslash, the results are unspecified.



20967        **-s**        Concatenate all of the lines of each separate input file in command line order. The  
 20968                      newline character of every line except the last line in each input file will be replaced  
 20969                      with the tab character, unless otherwise specified by the **-d** option.

## 20970 OPERANDS

20971        The following operand is supported:

20972        **file**        A pathname of an input file. If "-" is specified for one or more of the *files*, the standard  
 20973                      input will be used; the standard input will be read one line at a time, circularly, for each  
 20974                      instance of "-". Implementations support pasting of at least 12 *file* operands.

## 20975 STDIN

20976        The standard input will be used only if one or more *file* operands is "-". See the INPUT FILES  
 20977                      section.

## 20978 INPUT FILES

20979        The input files must be text files, except that line lengths will be unlimited.

## 20980 ENVIRONMENT VARIABLES

20981        The following environment variables affect the execution of *paste*:

20982        **LANG**        Provide a default value for the internationalisation variables that are unset or null. If  
 20983                      *LANG* is unset or null, the corresponding value from the implementation-dependent  
 20984                      default locale will be used. If any of the internationalisation variables contains an  
 20985                      invalid setting, the utility will behave as if none of the variables had been defined.

20986        **LC\_ALL**

20987                      If set to a non-empty string value, override the values of all the other  
 20988                      internationalisation variables.

20989        **LC\_CTYPE**

20990                      Determine the locale for the interpretation of sequences of bytes of text data as  
 20991                      characters (for example, single- as opposed to multi-byte characters in arguments and  
 20992                      input files).

20993        **LC\_MESSAGES**

20994                      Determine the locale that should be used to affect the format and contents of diagnostic  
 20995                      messages written to standard error.

20996 EX        **NLSPATH**

20997                      Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 20998 ASYNCHRONOUS EVENTS

20999        Default.

## 21000 STDOUT

21001        Concatenated lines of input files will be separated by the tab character (or other characters under  
 21002                      the control of the **-d** option) and terminated by a newline character.

## 21003 STDERR

21004        Used only for diagnostic messages.

## 21005 OUTPUT FILES

21006        None.

## 21007 EXTENDED DESCRIPTION

21008        None.

21009 **EXIT STATUS**

21010 The following exit values are returned:

21011 0 Successful completion.

21012 &gt;0 An error occurred.

21013 **CONSEQUENCES OF ERRORS**

21014 If one or more input files cannot be opened when the **-s** option is not specified, a diagnostic  
 21015 message will be written to standard error, but no output will be written to standard output. If  
 21016 the **-s** option is specified, the *paste* utility will provide the default behaviour described in Section  
 21017 1.9 on page 11.

21018 **APPLICATION USAGE**

21019 When the escape sequences of the *list* option-argument are used in a shell script, they must be  
 21020 quoted; otherwise, the shell treats the **\** as a special character.

21021 Portable applications should only use the specific backslash escaped delimiters presented in this  
 21022 specification. Historical implementations treat **\x**, where *x* is not in this list, as *x*, but future  
 21023 implementations are free to expand this list to recognise other common escapes similar to those  
 21024 accepted by *printf* and other standard utilities.

21025 Most of the standard utilities work on text files. The *cut* utility can be used to turn files with  
 21026 arbitrary line lengths into a set of text files containing the same data. The *paste* utility can be  
 21027 used to create (or recreate) files with arbitrary line lengths. For example, if **file** contains long  
 21028 lines:

21029 `cut -b 1-500 -n file > file1`21030 `cut -b 501- -n file > file2`

21031 creates **file1** (a text file) with lines no longer than 500 bytes (plus the newline character) and **file2**  
 21032 that contains the remainder of the data from **file**. Note that **file2** will not be a text file if there are  
 21033 lines in **file** that are longer than 500 + {**LINE\_MAX**} bytes. The original file can be recreated from  
 21034 **file1** and **file2** using the command:

21035 `paste -d "\0" file1 file2 > file`

21036 The commands:

21037 `paste -d "\0" ...`21038 `paste -d " " ...`

21039 are not necessarily equivalent; the latter is not specified by this specification and may result in  
 21040 an error. The construct **\0** is used to mean “no separator” because historical versions of *paste*  
 21041 did not follow the syntax guidelines, and the command:

21042 `paste -d" " ...`21043 could not be handled properly by *getopt()*.21044 **EXAMPLES**

21045 1. Write out a directory in four columns:

21046 `ls | paste - - - -`

21047 2. Combine pairs of lines from a file into single lines:

21048 `paste -s -d "\t\n" file`21049 **FUTURE DIRECTIONS**

21050 None.

21051 **SEE ALSO**21052 *cut, grep, pr.*21053 **CHANGE HISTORY**

21054 First released in Issue 2.

21055 **Issue 4**

21056 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 21057 NAME

21058 patch — apply changes to files

## 21059 SYNOPSIS

21060 patch [-blNR][ -c| -e| -n][ -d *dir*][ -D *define*][ -i *patchfile*]  
 21061 [-o *outfile*][ -p *num*][ -r *rejectfile*][ *file*]

## 21062 DESCRIPTION

21063 The *patch* utility reads a source (patch) file containing any of the three forms of difference (diff)  
 21064 listings produced by the *diff* utility (normal, context or in the style of *ed*) and apply those  
 21065 differences to a file. By default, *patch* reads from the standard input.

21066 The *patch* utility attempts to determine the type of the *diff* listing, unless overruled by a *-c*, *-e* or  
 21067 *-n* option.

21068 If the patch file contains more than one patch, *patch* will attempt to apply each of them as if they  
 21069 came from separate patch files. (In this case the name of the patch file must be determinable for  
 21070 each *diff* listing.)

## 21071 OPTIONS

21072 The *patch* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

21073 The following options are supported:

21074 **-b** Save a copy of the original contents of each modified file, before the differences are  
 21075 applied, in a file of the same name with the suffix **.orig** appended to it. If the file  
 21076 already exists, it will be overwritten; if multiple patches are applied to the same file, the  
 21077 **.orig** file will be written only for the first patch. When the *-o outfile* option is also  
 21078 specified, *file.orig* will not be created but, if *outfile* already exists, *outfile.orig* will be  
 21079 created.

21080 **-c** Interpret the patch file as a context difference (the output of the utility *diff* when the *-c*  
 21081 or *-C* options are specified).

21082 **-d dir** Change the current directory to *dir* before processing as described in the EXTENDED  
 21083 DESCRIPTION section.

21084 **-D define**

21085 Mark changes with the C preprocessor construct:

```
21086 #ifdef define
21087 ...
21088 #endif
```

21089 The option-argument *define* will be used as the differentiating symbol.

21090 **-e** Interpret the patch file as an *ed* script, rather than a *diff* script.

21091 **-i patchfile**

21092 Read the patch information from the file named by the pathname *patchfile*, rather than  
 21093 the standard input.

21094 **-l** (The letter ell.) Cause any sequence of blank characters in the difference script to match  
 21095 any sequence of blank characters in the input file. Other characters will be matched  
 21096 exactly.

21097 **-n** Interpret the script as a normal difference.

21098       **-N**       Ignore patches where the differences have already been applied to the file; by default,  
21099               already-applied patches are rejected.

21100       **-o outfile**  
21101               Instead of modifying the files (specified by the *file* operand or the difference listings)  
21102               directly, write a copy of the file referenced by each patch, with the appropriate  
21103               differences applied, to *outfile*. Multiple patches for a single file will be applied to the  
21104               intermediate versions of the file created by any previous patches, and will result in  
21105               multiple, concatenated versions of the file being written to *outfile*.

21106       **-p num**  
21107               For all pathnames in the patch file that indicate the names of files to be patched, delete  
21108               *num* pathname components from the beginning of each pathname. If the pathname in  
21109               the patch file is absolute, any leading slashes are considered the first component (that  
21110               is, **-p 1** removes the leading slashes). Specifying **-p 0** causes the full pathname to be  
21111               used. If **-p** is not specified, only the basename (the final pathname component) is used.

21112       **-R**       Reverse the sense of the patch script; that is, assume that the difference script was  
21113               created from the new version to the old version. The **-R** option cannot be used with *ed*  
21114               scripts. The *patch* utility attempts to reverse each portion of the script before applying  
21115               it. Rejected differences will be saved in swapped format. If this option is not specified,  
21116               and until a portion of the patch file is successfully applied, *patch* attempts to apply each  
21117               portion in its reversed sense as well as in its normal sense. If the attempt is successful,  
21118               the user will be prompted to determine if the **-R** option should be set.

21119       **-r rejectfile**  
21120               Override the default reject filename. In the default case, the reject file will have the  
21121               same name as the output file, with the suffix **.rej** appended to it. See **Patch Application**  
21122               on page 575.

## 21123 OPERANDS

21124       The following operand is supported:

21125       *file*       A pathname of a file to patch.

## 21126 STDIN

21127       See the INPUT FILES section.

## 21128 INPUT FILES

21129       Input files must be text files.

## 21130 ENVIRONMENT VARIABLES

21131       The following environment variables affect the execution of *patch*:

21132       **LANG**       Provide a default value for the internationalisation variables that are unset or null. If  
21133               **LANG** is unset or null, the corresponding value from the implementation-dependent  
21134               default locale will be used. If any of the internationalisation variables contains an  
21135               invalid setting, the utility will behave as if none of the variables had been defined.

21136       **LC\_ALL**  
21137               If set to a non-empty string value, override the values of all the other  
21138               internationalisation variables.

21139       **LC\_CTYPE**  
21140               Determine the locale for the interpretation of sequences of bytes of text data as  
21141               characters (for example, single- as opposed to multi-byte characters in arguments and  
21142               input files).

|          |                                                                                                                                   |
|----------|-----------------------------------------------------------------------------------------------------------------------------------|
| 21143    | <b>LC_MESSAGES</b>                                                                                                                |
| 21144    | Determine the locale that should be used to affect the format and contents of diagnostic                                          |
| 21145    | messages written to standard error and informative messages written to standard                                                   |
| 21146    | output.                                                                                                                           |
| 21147 EX | <b>NLSPATH</b>                                                                                                                    |
| 21148    | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                                           |
| 21149    | <b>LC_TIME</b>                                                                                                                    |
| 21150    | Determine the locale for recognising the format of file timestamps written by the <i>diff</i>                                     |
| 21151    | utility in a context-difference input file.                                                                                       |
| 21152    | <b>ASYNCHRONOUS EVENTS</b>                                                                                                        |
| 21153    | Default.                                                                                                                          |
| 21154    | <b>STDOUT</b>                                                                                                                     |
| 21155    | Not used.                                                                                                                         |
| 21156    | <b>STDERR</b>                                                                                                                     |
| 21157    | Used for diagnostic and informational messages.                                                                                   |
| 21158    | <b>OUTPUT FILES</b>                                                                                                               |
| 21159    | The output of the <i>patch</i> utility, the save files ( <b>.orig</b> suffixes) and the reject files ( <b>.rej</b> suffixes) will |
| 21160    | be text files.                                                                                                                    |
| 21161    | <b>EXTENDED DESCRIPTION</b>                                                                                                       |
| 21162    | A patchfile may contain patching instructions for more than one file; filenames are determined                                    |
| 21163    | as specified in <b>Filename Determination</b> on page 575. When the <b>-b</b> option is specified, for each                       |
| 21164    | patched file, the original will be saved in a file of the same name with the suffix <b>.orig</b> appended                         |
| 21165    | to it.                                                                                                                            |
| 21166    | For each patched file, a reject file may also be created as noted in <b>Patch Application</b> on page 575.                        |
| 21167    | In the absence of a <b>-r</b> option, the name of this file will be formed by appending the suffix <b>.rej</b> to                 |
| 21168    | the original filename.                                                                                                            |
| 21169    | <b>Patchfile Format</b>                                                                                                           |
| 21170    | The patch file must contain zero or more lines of header information followed by one or more                                      |
| 21171    | patches. Each patch must contain zero or more lines of filename identification in the format                                      |
| 21172    | produced by <i>diff -c</i> , and one or more sets of <i>diff</i> output, which are customarily called hunks.                      |
| 21173    | The <i>patch</i> utility recognises the following expression in the header information:                                           |
| 21174    | <b>Index:</b> <i>pathname</i>                                                                                                     |
| 21175    | The file to be patched is named <i>pathname</i> .                                                                                 |
| 21176    | If all lines (including headers) within a patch begin with the same leading sequence of blank                                     |
| 21177    | characters, the <i>patch</i> utility will remove this sequence before proceeding. Within each patch, if                           |
| 21178    | the type of difference is context, the <i>patch</i> utility recognises the following expressions:                                 |
| 21179    | <b>***</b> <i>filename timestamp</i>                                                                                              |
| 21180    | The patches arose from <i>filename</i> .                                                                                          |
| 21181    | <b>---</b> <i>filename timestamp</i>                                                                                              |
| 21182    | The patches should be applied to <i>filename</i> .                                                                                |
| 21183    | Each hunk within a patch must be the <i>diff</i> output to change a line range within the original file.                          |
| 21184    | The line numbers for successive hunks within a patch must occur in ascending order.                                               |

21185 **Filename Determination**

21186 If no *file* operand is specified, *patch* performs the following steps to obtain a pathname:

- 21187 1. If the patch contains the strings **\*\*\*** and **---**, the *patch* utility strips components from the  
21188 beginning of each pathname (depending on the presence or value of the **-p** option), then  
21189 tests for the existence of both files in the current directory (or directory specified with the  
21190 **-d** option). If both files exist, *patch* assumes that no pathname can be obtained from this  
21191 step.
- 21192 2. If the header information contains a line with the string **Index:**, *patch* utility strips  
21193 components from the beginning of the pathname (depending on **-p**), then tests for the  
21194 existence of this file in the current directory (or directory specified with the **-d** option).
- 21195 EX 3. If an **SCCS** directory exists in the current directory, *patch* will attempt to perform a *get -e*  
21196 *SCCS/s.filename* command to retrieve an editable version of the file.
- 21197 4. If no pathname can be obtained by applying the previous steps, or if the pathnames  
21198 obtained do not exist, *patch* will write a prompt to standard output and request a filename  
21199 interactively from standard input.

21200 **Patch Application**

21201 If the **-c**, **-e** or **-n** option is present, the *patch* utility will interpret information within each hunk  
21202 as a context difference, an *ed* difference or a normal difference, respectively. In the absence of  
21203 any of these options, the *patch* utility determines the type of difference based on the format of  
21204 information within the hunk.

21205 For each hunk, the *patch* utility begins to search for the place to apply the patch at the line  
21206 number at the beginning of the hunk, plus or minus any offset used in applying the previous  
21207 hunk. If lines matching the hunk context are not found, *patch* scans both forwards and  
21208 backwards at least 1000 bytes for a set of lines that match the hunk context.

21209 If no such place is found and it is a context difference, then another scan will take place, ignoring  
21210 the first and last line of context. If that fails, the first two and last two lines of context will be  
21211 ignored and another scan will be made. Implementations may search more extensively for  
21212 installation locations.

21213 If no location can be found, the *patch* utility will append the hunk to the reject file. The rejected  
21214 hunk will be written in context-difference format regardless of the format of the patch file. If the  
21215 input was a normal or *ed*-style difference, the reject file may contain differences with zero lines  
21216 of context. The line numbers on the hunks in the reject file may be different from the line  
21217 numbers in the patch file since they will reflect the approximate locations for the failed hunks in  
21218 the new file rather than the old one.

21219 If the type of patch is an *ed* diff, the implementation may accomplish the patching by invoking  
21220 the *ed* utility.

21221 **EXIT STATUS**

21222 The following exit values are returned:

- 21223 0 Successful completion.
- 21224 1 One or more lines were written to a reject file.
- 21225 >1 An error occurred.

21226 **CONSEQUENCES OF ERRORS**

21227 Patches that cannot be correctly placed in the file will be written to a reject file.

21228 **APPLICATION USAGE**

21229       The **-R** option will not work with *ed* scripts because there is too little information to reconstruct  
21230       the reverse operation.

21231       The **-p** option makes it possible to customise a patchfile to local user directory structures  
21232       without manually editing the patchfile. For example, if the filename in the patch file was:

21233             /curds/whey/src/blurfl/blurfl.c

21234       Setting **-p 0** gives the entire pathname unmodified; **-p 1** gives:

21235             curds/whey/src/blurfl/blurfl.c

21236       without the leading slash, **-p 4** gives:

21237             blurfl/blurfl.c

21238       and not specifying **-p** at all gives:

21239             blurfl.c .

21240       When using **-b** in some file system implementations, the saving of a **.orig** file may produce  
21241       unwanted results. In the case of 12, 13 or 14-character filenames, on file systems supporting 14-  
21242       character maximum filenames, the **.orig** file will overwrite the new file.

21243 **EXAMPLES**

21244       None.

21245 **FUTURE DIRECTIONS**

21246       The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
21247       interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
21248       corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
21249       finalised.

21250 **SEE ALSO**

21251       *ed*, *diff*.

21252 **CHANGE HISTORY**

21253       First released in Issue 4.

21254 **Issue 5**

21255       FUTURE DIRECTIONS section added.



21256 **NAME**

21257 pathchk — check pathnames

21258 **SYNOPSIS**21259 pathchk [-p] *pathname...*21260 **DESCRIPTION**

21261 The *pathchk* utility will check that one or more pathnames are valid (that is, they could be used to  
 21262 access or create a file without causing syntax errors) and portable (that is, no filename truncation  
 21263 will result). More extensive portability checks are provided by the **-p** option.

21264 By default, the *pathchk* utility will check each component of each *pathname* operand based on the  
 21265 underlying file system. A diagnostic will be written for each *pathname* operand that:

- 21266 • is longer than {PATH\_MAX} bytes (see **Pathname Variable Values** in the **XSH** specification  
 21267 <**limits.h**> description)
- 21268 • contains any component longer than {NAME\_MAX} bytes in its containing directory
- 21269 • contains any component in a directory that is not searchable
- 21270 • contains any character in any component that is not valid in its containing directory.

21271 The format of the diagnostic message is not specified, but will indicate the error detected and the  
 21272 corresponding *pathname* operand.

21273 It will not be considered an error if one or more components of a *pathname* operand do not exist  
 21274 as long as a file matching the pathname specified by the missing components could be created  
 21275 that does not violate any of the checks specified above.

21276 **OPTIONS**21277 The *pathchk* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

21278 The following option is supported:

- 21279 **-p** Instead of performing checks based on the underlying file system, write a diagnostic for  
 21280 each *pathname* operand that:
  - 21281 • is longer than {\_POSIX\_PATH\_MAX} bytes (see **Minimum Values** in the **XSH**  
 21282 specification <**limits.h**> description)
  - 21283 • contains any component longer than {\_POSIX\_NAME\_MAX} bytes
  - 21284 • contains any character in any component that is not in the portable filename  
 21285 character set.

21286 **OPERANDS**

21287 The following operand is supported:

- 21288 *pathname*  
 21289 A pathname to be checked.

21290 **STDIN**

21291 Not used.

21292 **INPUT FILES**

21293 None.

21294 **ENVIRONMENT VARIABLES**

21295 The following environment variables affect the execution of *pathchk*:

21296 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 21297 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 21298 default locale will be used. If any of the internationalisation variables contains an  
 21299 invalid setting, the utility will behave as if none of the variables had been defined.

21300 **LC\_ALL**

21301 If set to a non-empty string value, override the values of all the other  
 21302 internationalisation variables.

21303 **LC\_CTYPE**

21304 Determine the locale for the interpretation of sequences of bytes of text data as  
 21305 characters (for example, single- as opposed to multi-byte characters in arguments).

21306 **LC\_MESSAGES**

21307 Determine the locale that should be used to affect the format and contents of diagnostic  
 21308 messages written to standard error.

21309 EX **NLSPATH**

21310 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

21311 **ASYNCHRONOUS EVENTS**

21312 Default.

21313 **STDOUT**

21314 Not used.

21315 **STDERR**

21316 Used only for diagnostic messages.

21317 **OUTPUT FILES**

21318 None.

21319 **EXTENDED DESCRIPTION**

21320 None.

21321 **EXIT STATUS**

21322 The following exit values are returned:

21323 0 All *pathname* operands passed all of the checks.

21324 >0 An error occurred.

21325 **CONSEQUENCES OF ERRORS**

21326 Default.

21327 **APPLICATION USAGE**

21328 The *test* utility can be used to determine if a given pathname names an existing file; it will not,  
 21329 however, give any indication of whether or not any component of the pathname was truncated  
 21330 in a directory where the {\_POSIX\_NO\_TRUNC} feature is not in effect. The *pathchk* utility does  
 21331 not check for file existence; it performs checks to determine if a pathname does exist or could be  
 21332 created with no pathname component truncation.

21333 The *noclobber* option in the shell (see the *set* special built-in) can be used to atomically create a  
 21334 file. As with all file creation semantics in the **XSH** specification, it guarantees atomic creation,  
 21335 but still depends on applications to agree on conventions and cooperate on the use of files after  
 21336 they have been created.

21337 **EXAMPLES**

21338 To verify that all pathnames in an imported data interchange archive are legitimate and  
 21339 unambiguous on the current system:

```
21340 pax -f archive | sed -e '/ == ./s///' | xargs pathchk
21341 if [$? -eq 0]
21342 then
21343 pax -r -f archive
21344 else
21345 echo Investigate problems before importing files.
21346 exit 1
21347 fi
```

21348 To verify that all files in the current directory hierarchy could be moved to any system  
 21349 conforming to the **XSH** specification that also supports the *pax* utility:

```
21350 find . -print | xargs pathchk -p
21351 if [$? -eq 0]
21352 then
21353 pax -w -f archive .
21354 else
21355 echo Portable archive cannot be created.
21356 exit 1
21357 fi
```

21358 To verify that a user-supplied pathname names a readable file and that the application can create  
 21359 a file extending the given path without truncation and without overwriting any existing file:

```
21360 case $- in
21361 *C*) reset="";;
21362 *) reset="set +C"
21363 set -C;;
21364 esac
21365 test -r "$path" && pathchk "$path.out" &&
21366 rm "$path.out" > "$path.out"
21367 if [$? -ne 0]; then
21368 printf "%s: %s not found or %s.out fails \
21369 creation checks.\n" $0 "$path" "$path"
21370 $reset # reset the noclobber option in case a trap
21371 # on EXIT depends on it
21372 exit 1
21373 fi
21374 $reset
21375 PROCESSING < "$path" > "$path.out"
```

21376 The following assumptions are made in this example:

- 21377 1. **PROCESSING** represents the code that will be used by the application to use **\$path** once it  
 21378 is verified that **\$path.out** will work as intended.
- 21379 2. The state of the *noclobber* option is unknown when this code is invoked and should be set  
 21380 on exit to the state it was in when this code was invoked. (The **reset** variable is used in this  
 21381 example to restore the initial state.)

- 21382           3. Note the usage of:
- 21383                 rm "\$path.out" > "\$path.out"
- 21384           a. The *pathchk* command has already verified, at this point, that **\$path.out** will not be
- 21385                 truncated.
- 21386           b. With the *noclobber* option set, the shell will verify that **\$path.out** does not already
- 21387                 exist before invoking *rm*.
- 21388           c. If the shell succeeded in creating **\$path.out**, *rm* will remove it so that the application
- 21389                 can create the file again in the **PROCESSING** step.
- 21390           d. If the **PROCESSING** step wants the file to exist already when it is invoked, the:
- 21391                 rm "\$path.out" > "\$path.out"
- 21392                 should be replaced with:
- 21393                 > "\$path.out"
- 21394                 which will verify that the file did not already exist, but leave **\$path.out** in place for
- 21395                 use by **PROCESSING**.
- 21396 **FUTURE DIRECTIONS**
- 21397                 None.
- 21398 **SEE ALSO**
- 21399                 *test*, Section 2.7 on page 40.
- 21400 **CHANGE HISTORY**
- 21401                 First released in Issue 4.

21402 **NAME**

21403       pax — portable archive interchange

21404 **SYNOPSIS**21405       pax [-cdnv][*-f archive*][*-s replstr*]...[*pattern*...]21406       pax -r[-cdiknuv][*-f archive*][*-o options*]...[*-p string*]...  
21407       [*-s replstr*]...[*pattern*...]21408       pax -w[-dituvX][*-b blocksize*][*-a*][*-f archive*][*-o options*]...  
21409       [*-s replstr*]...[*-x format*][*file*...]21410       pax -r -w[-diklntuvX][*-p string*]...[*-s replstr*]...[*file*...]  
21411       *directory*21412 **DESCRIPTION**21413       The *pax* utility reads, writes and writes lists of the members of archive files and copy directory  
21414       hierarchies. A variety of archive formats are supported; see the *-x format* option.21415       The action to be taken depends on the presence of the *-r* and *-w* options. The four combinations  
21416       of *-r* and *-w* are referred to as the four modes of operation: *list*, *read*, *write* and *copy* modes,  
21417       corresponding respectively to the four forms shown in the SYNOPSIS section.21418       *list*       In list mode (when neither *-r* nor *-w* are specified), *pax* writes the names of the  
21419       members of the archive file read from the standard input, with pathnames matching  
21420       the specified patterns, to standard output. If a named file is of type directory, the file  
21421       hierarchy rooted at that file will be written out as well.21422       *read*       In read mode (when *-r* is specified, but *-w* is not), *pax* extracts the members of the  
21423       archive file read from the standard input, with pathnames matching the specified  
21424       patterns. If an extracted file is of type directory, the file hierarchy rooted at that file will  
21425       be extracted as well. The extracted files is created relative to the current file hierarchy.21426       The ownership, access and modification times, and file mode of the restored files are  
21427       discussed under the *-p* option.21428       *write*       In write mode (when *-w* is specified, but *-r* is not), *pax* writes the contents of the file  
21429       operands to the standard output in an archive format. If no *file* operands are specified,  
21430       a list of files to copy, one per line, will be read from the standard input. A file of type  
21431       directory will include all of the files in the file hierarchy rooted at the file.21432       *copy*       In copy mode (when both *-r* and *-w* are specified), *pax* copies the file operands to the  
21433       destination directory.21434       If no *file* operands are specified, a list of files to copy, one per line, will be read from the  
21435       standard input. A file of type directory will include all of the files in the file hierarchy  
21436       rooted at the file.21437       The effect of the copy is as if the copied files were written to an archive file and then  
21438       subsequently extracted, except that there may be hard links between the original and  
21439       the copied files. If the destination directory is a subdirectory of one of the files to be  
21440       copied, the results are unspecified. If the destination directory is a file of a type not  
21441       defined by the XSH specification, the results are implementation-dependent; otherwise  
21442       it is an error for the file named by the directory operand not to exist, not be writable by  
21443       the user, or not be a file of type directory.

21444 In read or copy modes, if intermediate directories are necessary to extract an archive member,  
 21445 *pax* will perform actions equivalent to the **XSH** specification *mkdir()* function, called with the  
 21446 following arguments:

- 21447 • the intermediate directory used as the *path* argument
- 21448 • the value of the bitwise inclusive OR of S\_IRWXU, S\_IRWXG and S\_IRWXO as the *mode*  
 21449 argument.

21450 If any specified *pattern* or *file* operands are not matched by at least one file or archive member,  
 21451 *pax* will write a diagnostic message to standard error for each one that did not match and exit  
 21452 with a non-zero exit status.

21453 The supported archive formats are automatically detected on input. The default output archive  
 21454 format is implementation-dependent.

21455 A single archive can span multiple files. The *pax* utility determines, in an implementation-  
 21456 dependent manner, what file to read or write as the next file.

21457 If the selected archive format supports the specification of linked files, it is an error if these files  
 21458 cannot be linked when the archive is extracted. Any of the various names in the archive that  
 21459 represent a file can be used to select the file for extraction.

## 21460 OPTIONS

21461 The *pax* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 21462 that the order of presentation of the **-s** options is significant.

21463 The following options are supported:

- 21464 **-r** Read an archive file from standard input.
- 21465 **-w** Write files to the standard output in the specified archive format.
- 21466 **-a** Append files to the end of the archive. It is implementation-dependent which devices  
 21467 on the system support appending. Additional file formats unspecified by this  
 21468 specification may impose restrictions on appending.
- 21469 **-b *blocksize***  
 21470 Block the output at a positive decimal integer number of bytes per write to the archive  
 21471 file. Devices and archive formats may impose restrictions on blocking. Blocking is  
 21472 automatically determined on input. Portable applications must not specify a *blocksize*  
 21473 value larger than 32 256. Default blocking when creating archives depends on the  
 21474 archive format. (See the **-x** option below.)
- 21475 **-c** Match all file or archive members except those specified by the *pattern* or *file* operands.
- 21476 **-d** Cause files of type directory being copied or archived or archive members of type  
 21477 directory being extracted to match only the file or archive member itself and not the file  
 21478 hierarchy rooted at the file.
- 21479 **-f *archive***  
 21480 Specify the pathname of the input or output archive, overriding the default standard  
 21481 input (in list or read modes) or standard output (write mode).
- 21482 **-i** Interactively rename files or archive members. For each archive member matching a  
 21483 *pattern* operand or file matching a *file* operand, a prompt will be written to the file  
 21484 **/dev/tty**. The prompt will contain the name of the file or archive member, but the  
 21485 format is otherwise unspecified. A line will then be read from **/dev/tty**. If this line is  
 21486 blank, the file or archive member will be skipped. If this line consists of a single period,  
 21487 the file or archive member will be processed with no modification to its name.

21488 Otherwise, its name will be replaced with the contents of the line. The *pax* utility will  
 21489 immediately exit with a non-zero exit status if end-of-file is encountered when reading  
 21490 a response or if **/dev/tty** cannot be opened for reading and writing.

21491 **-k** Prevent the overwriting of existing files.

21492 **-l** (The letter ell.) Link files. In copy mode, hard links will be made between the source  
 21493 and destination file hierarchies whenever possible.

21494 **-n** Select the first archive member that matches each *pattern* operand. No more than one  
 21495 archive member will be matched for each pattern (although members of type directory  
 21496 will still match the file hierarchy rooted at that file).

21497 **-o options**  
 21498 Provide information to the implementation to modify the algorithm for extracting or  
 21499 writing files that is specific to the file format specified by **-x**. This issue does not  
 21500 specify any such options and a portable application must not use the **-o** option.

21501 **-p string**  
 21502 Specify one or more file characteristic options (privileges). The *string* option-argument  
 21503 must be a string specifying file characteristics to be retained or discarded on extraction.  
 21504 The string consists of the specification characters a, e, m, o and p. Other,  
 21505 implementation-dependent, characters can be included. Multiple characteristics can be  
 21506 concatenated within the same string and multiple **-p** options can be specified. The  
 21507 meaning of the specification characters are as follows:

21508 a Do not preserve file access times.

21509 e Preserve the user ID, group ID, file mode bits (see **file mode bits** in the **XBD**  
 21510 specification, **Chapter 2, Glossary**), access time, modification time and any other,  
 21511 implementation-dependent, file characteristics.

21512 m Do not preserve file modification times.

21513 o Preserve the user ID and group ID.

21514 p Preserve the file mode bits. Other, implementation-dependent file-mode attributes  
 21515 may be preserved.

21516 In the preceding list, “preserve” indicates that an attribute stored in the archive will be  
 21517 given to the extracted file, subject to the permissions of the invoking process;  
 21518 otherwise, the attribute will be determined as part of the normal file creation action.

21519 If neither the e nor the o specification character is specified, or the user ID and group ID  
 21520 are not preserved for any reason, *pax* will not set the S\_ISUID and S\_ISGID bits of the  
 21521 file mode.

21522 If the preservation of any of these items fails for any reason, *pax* will write a diagnostic  
 21523 message to standard error. Failure to preserve these items will affect the final exit  
 21524 status, but will not cause the extracted file to be deleted.

21525 If file-characteristic letters in any of the *string* option-arguments are duplicated or  
 21526 conflict with each other, the ones given last will take precedence. For example, if  
 21527 **-p eme** is specified, file modification times will be preserved.

21528       **-s replstr**  
 21529           Modify file or archive member names named by *pattern* or *file* operands according to  
 21530           the substitution expression *replstr*, using the syntax of the *ed* utility. The concepts of  
 21531           “address” and “line” are meaningless in the context of the *pax* utility, and must not be  
 21532           supplied. The format is:

21533                 -s /old/new/[gp]

21534           where as in *ed*, *old* is a basic regular expression and *new* can contain an ampersand, \n  
 21535           (where *n* is a digit) backreferences, or subexpression matching. The *old* string also is  
 21536           permitted to contain newline characters.

21537           Any non-null character can be used as a delimiter (/ shown here). Multiple **-s**  
 21538           expressions can be specified; the expressions will be applied in the order specified,  
 21539           terminating with the first successful substitution. The optional trailing *g* is as defined  
 21540           in the *ed* utility. The optional trailing *p* causes successful substitutions to be written to  
 21541           standard error. File or archive member names that substitute to the empty string are  
 21542           ignored when reading and writing archives.

21543       **-t**       Cause the access times of the archived files to be the same as they were before being  
 21544           read by *pax*.

21545       **-u**       Ignore files that are older (having a less recent file modification time) than a pre-  
 21546           existing file or archive member with the same name. In read mode, an archive member  
 21547           with the same name as a file in the file system will be extracted if the archive member is  
 21548           newer than the file. In write mode, an archive file member with the same name as a file  
 21549           in the file system will be superseded if the file is newer than the archive member. It is  
 21550           unspecified if this is accomplished by actual replacement in the archive or by  
 21551           appending to the archive. In copy mode, the file in the destination hierarchy will be  
 21552           replaced by the file in the source hierarchy or by a link to the file in the source hierarchy  
 21553           if the file in the source hierarchy is newer.

21554       **-v**       In list mode, produce a verbose table of contents (see the STDOUT section). Otherwise,  
 21555           write archive member pathnames to standard error (see the STDERR section).

21556       **-x format**  
 21557           Specify the output archive format. The *pax* utility recognises the following formats:

21558       **cpio**      The extended *cpio* interchange format; see the EXTENDED DESCRIPTION  
 21559           section. The default *blocksize* for this format for character special archive files  
 21560           is 5120. Implementations support all *blocksize* values less than or equal to  
 21561           32 256 that are multiples of 512.

21562       **ustar**     The extended *tar* interchange format; see the EXTENDED DESCRIPTION  
 21563           section. The default *blocksize* for this format for character special archive files  
 21564           is 10 240. Implementations support all *blocksize* values less than or equal to  
 21565           32 256 that are multiples of 512.

21566           Implementation-dependent formats will specify a default block size as well as any  
 21567           other block sizes supported for character special archive files.

21568           Any attempt to append to an archive file in a format different from the existing archive  
 21569           format will cause *pax* to exit immediately with a non-zero exit status.

21570       **-X**       When traversing the file hierarchy specified by a pathname, *pax* will not descend into  
 21571           directories that have a different device ID (**st\_dev**, see the XSH specification *stat()*).

21572       The options that operate on the names of files or archive members (**-c**, **-i**, **-n**, **-s**, **-u** and **-v**)  
 21573       interact as follows. In read mode, the archive members are selected based on the user-specified



21574 *pattern* operands as modified by the **-c**, **-n** and **-u** options. Then, any **-s** and **-i** options will  
 21575 modify, in that order, the names of the selected files. The **-v** option will write names resulting  
 21576 from these modifications.

21577 In write mode, the files are selected based on the user-specified pathnames as modified by the  
 21578 **-n** and **-u** options. Then, any **-s** and **-i** options will, in that order, modify the names of these  
 21579 selected files. The **-v** option will write names resulting from these modifications.

21580 If both the **-u** and **-n** options are specified, *pax* does not consider a file selected unless it is newer  
 21581 than the file to which it is compared.

## 21582 OPERANDS

21583 The following operands are supported:

21584 *directory*

21585 The destination directory pathname for copy mode.

21586 *file*

A pathname of a file to be copied or archived.

21587 *pattern* A pattern matching one or more pathnames of archive members. A pattern must be  
 21588 given in the name-generating notation of the pattern matching notation in Section 2.13  
 21589 on page 64, including the filename expansion rules in Section 2.13.3 on page 66. The  
 21590 default, if no *pattern* is specified, is to select all members in the archive.

## 21591 STDIN

21592 In write mode, the standard input is used only if no *file* operands are specified. It must be a text  
 21593 file containing a list of pathnames, one per line, without leading or trailing blank characters.

21594 In list and read modes, the standard input must be an archive file.

21595 Otherwise, the standard input will not be used.

## 21596 INPUT FILES

21597 The input file named by the *archive* option-argument, or standard input when the archive is read  
 21598 from there, will be a file formatted according to one of the specifications in the EXTENDED  
 21599 DESCRIPTION section or some other, implementation-dependent, format.

21600 The file **/dev/tty** is used to write prompts and read responses.

## 21601 ENVIRONMENT VARIABLES

21602 The following environment variables affect the execution of *pax*:

21603 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 21604 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 21605 default locale will be used. If any of the internationalisation variables contains an  
 21606 invalid setting, the utility will behave as if none of the variables had been defined.

21607 **LC\_ALL**

21608 If set to a non-empty string value, override the values of all the other  
 21609 internationalisation variables.

21610 **LC\_COLLATE**

21611 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 21612 character collating elements used in the pattern matching expressions for the *pattern*  
 21613 operand, the basic regular expression for the **-s** option, and the extended regular  
 21614 expression defined for the **yesexpr** locale keyword in the LC\_MESSAGES category.

21615 **LC\_CTYPE**

21616 Determine the locale for the interpretation of sequences of bytes of text data as  
 21617 characters (for example, single- as opposed to multi-byte characters in arguments and

21618 input files), the behaviour of character classes used in the extended regular expression  
 21619 defined for the **yesexpr** locale keyword in the LC\_MESSAGES category, and pattern  
 21620 matching.

#### 21621 **LC\_MESSAGES**

21622 Determine the locale for the processing of affirmative responses that should be used to  
 21623 affect the format and contents of diagnostic messages written to standard error.

#### 21624 **LC\_TIME**

21625 Determine the format and contents of date and time strings when the **-v** option is  
 21626 specified.

#### 21627 EX **NLSPATH**

21628 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

### 21629 **ASYNCHRONOUS EVENTS**

21630 Default.

### 21631 **STDOUT**

21632 In write mode, if **-f** is not specified, the standard output will be the archive formatted according  
 21633 to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
 21634 implementation-dependent format. (See **-x format**.)

21635 In list mode, the table of contents of the selected archive members will be written to standard  
 21636 output using the following format:

21637 "%s\n", <pathname>

21638 If the **-v** option is specified in list mode, the table of contents of the selected archive members  
 21639 will be written to standard output using the following formats:

21640 For pathnames representing hard links to previous members of the archive:

21641 "%sΔ==Δ%s\n", <ls -l listing>, <linkname>

21642 For all other pathnames:

21643 "%s\n", <ls -l listing>

21644 where <ls -l listing> is the format specified by the **ls** utility with the **-l** option. When writing  
 21645 pathnames in this format, it is unspecified what is written for fields for which the underlying  
 21646 archive format does not have the correct information, although the correct number of blank-  
 21647 character-separated fields will be written.

21648 In list mode, standard output will not be buffered more than a line at a time.

### 21649 **STDERR**

21650 If **-v** is specified in read, write or copy modes, **pax** will write the pathnames it processes to the  
 21651 standard error output using the following format:

21652 "%s\n", <pathname>

21653 These pathnames will be written as soon as processing is begun on the file or archive member,  
 21654 and will be flushed to standard error. The trailing newline character, which will not be buffered,  
 21655 will be written when the file has been read or written.

21656 If the **-s** option is specified, and the replacement string has a trailing **p**, substitutions will be  
 21657 written to standard error in the following format:

21658 "%sΔ>>Δ%s\n", <original pathname>, <new pathname>

21659 In all operating modes of *pax*, optional messages of unspecified format concerning the input  
 21660 archive format and volume number, the number of files, blocks, volumes and media parts as  
 21661 well as other diagnostic messages may be written to standard error.

21662 In all formats, for both standard output and standard error, it is unspecified how non-printable  
 21663 characters in pathnames or linknames are written.

## 21664 OUTPUT FILES

21665 In read mode, the extracted or copied output files are of the archived file type.

21666 In write mode, the output file named by the *-f* option-argument is a file formatted according to  
 21667 one of the specifications in the EXTENDED DESCRIPTION section, or some other,  
 21668 implementation-dependent format.

## 21669 EXTENDED DESCRIPTION

### 21670 Extended *cpio* Format

21671 The octet-oriented *cpio* archive format is a series of entries, each comprising a header that  
 21672 describes the file, the name of the file and then the contents of the file.

21673 An archive may be recorded as a series of fixed-size blocks of octets. This blocking will be used  
 21674 only to make physical I/O more efficient. The last group of blocks always will be at the full size.

21675 For the octet-oriented *cpio* archive format, the individual entry information will be in the order  
 21676 indicated and described by the following table:  
 21677

| Header Field Name    | Length (in Octets) | Interpreted as  |
|----------------------|--------------------|-----------------|
| <b>c_magic</b>       | 6                  | Octal number    |
| <b>c_dev</b>         | 6                  | Octal number    |
| <b>c_ino</b>         | 6                  | Octal number    |
| <b>c_mode</b>        | 6                  | Octal number    |
| <b>c_uid</b>         | 6                  | Octal number    |
| <b>c_gid</b>         | 6                  | Octal number    |
| <b>c_nlink</b>       | 6                  | Octal number    |
| <b>c_rdev</b>        | 6                  | Octal number    |
| <b>c_mtime</b>       | 11                 | Octal number    |
| <b>c_namesize</b>    | 6                  | Octal number    |
| <b>c_filesize</b>    | 11                 | Octal number    |
| Filename Field Name  | Length             | Interpreted as  |
| <b>c_name</b>        | c_namesize         | Pathname string |
| File Data Field Name | Length             | Interpreted as  |
| <b>c_filedata</b>    | c_filesize         | Data            |

21694 **Table 3-11** Octet-oriented *cpio* Archive Entry

### 21695 The *cpio* Header

21696 For each file in the archive, a header as defined previously will be written. The information in  
 21697 the header fields will be written as streams of the ISO/IEC 646:1991 standard characters  
 21698 interpreted as octal numbers. The octal numbers will be extended to the necessary length by  
 21699 appending the ISO/IEC 646:1991 standard IRV zeros at the most-significant-digit end of the  
 21700 number; the result is written to the most-significant-digit of the stream of octets first. The fields  
 21701 are interpreted as follows:

21702 **c\_magic**  
 21703 Identify the archive as being a transportable archive by containing the identifying value  
 21704 "070707".

21705 **c\_dev**  
 21706 **c\_ino** Contains values that uniquely identify the file within the archive (that is, no files will  
 21707 contain the same pair of **c\_dev** and **c\_ino** values unless they are links to the same file).  
 21708 The values will be determined in an unspecified manner.

21709 **c\_mode** Contains the file type and access permissions as defined in the following table:  
 21710  
 21711

| File Permissions Name | Value    | Indicates               |
|-----------------------|----------|-------------------------|
| C_IRUSR               | 000 400  | Read by owner.          |
| C_IWUSR               | 000 200  | Write by owner.         |
| C_IXUSR               | 000 100  | Execute by owner.       |
| C_IRGRP               | 000 040  | Read by group.          |
| C_IWGRP               | 000 020  | Write by group.         |
| C_IXGRP               | 000 010  | Execute by group.       |
| C_IROTH               | 000 004  | Read by others.         |
| C_IWOTH               | 000 002  | Write by others.        |
| C_IXOTH               | 000 001  | Execute by others.      |
| C_ISUID               | 004 000  | Set <i>uid</i> .        |
| C_ISGID               | 002 000  | Set <i>gid</i> .        |
| C_ISVTX               | 001 000  | Reserved.               |
| File Type Name        | Value    | Indicates               |
| C_ISDIR               | 040 000  | Directory.              |
| C_ISFIFO              | 010 000  | FIFO.                   |
| C_ISREG               | 0100 000 | Regular file.           |
| C_ISBLK               | 060 000  | Block special file.     |
| C_ISCHR               | 020 000  | Character special file. |
| C_ISCTG               | 0110 000 | Reserved.               |
| C_ISLNK               | 0120 000 | Reserved.               |
| C_ISSOCK              | 0140 000 | Reserved.               |

21724  
 21725  
 21726  
 21727  
 21728  
 21729  
 21730  
 21731  
 21732  
 21733 **Table 3-12** Values for *cpio* **c\_mode** Field

21734 Directories, FIFOs and regular files are supported on a system conforming to this  
 21735 specification; additional values defined previously are reserved for compatibility with  
 21736 existing systems. Additional file types may be supported; however, such files should  
 21737 not be written to archives intended to be transported to other systems.

21738 **c\_uid** Contains the user ID of the owner.

21739 **c\_gid** Contains the group ID of the group.

21740 **c\_nlink** Contains the number of links referencing the file at the time the archive was created.

21741 **c\_rdev** Contains implementation-dependent information for character or block special files.

21742 **c\_mtime**  
 21743 Contains the latest time of modification of the file at the time the archive was created.

21744 **c\_namesize**  
 21745 Contains the length of the pathname, including the terminating NUL character.

21746       **c\_filesize**  
 21747           Contains the length of the file in octets. This will be the length of the data section  
 21748           following the header structure.

### 21749       **The cpio Filename**

21750       The **c\_name** field contains the pathname of the file. The length of this field in octets is the value  
 21751       of **c\_namesize**. If a filename is found on the medium that would create an invalid pathname, it is  
 21752       implementation-dependent if the data from the file is stored on the file hierarchy and under  
 21753       what name it is stored.

21754       All characters are represented in the ISO/IEC 646:1991 standard IRV. For maximum portability  
 21755       between implementations, names should be selected from characters represented by the  
 21756       portable filename character set as octets with the most significant bit zero. If an implementation  
 21757       supports the use of characters outside the portable filename character set in names for files,  
 21758       users and groups, one or more implementation-dependent encodings of these characters will be  
 21759       provided for interchange purposes. However, the *pax* utility will never create filenames on the  
 21760       local system that cannot be accessed via the procedures described previously in this  
 21761       specification. If a filename is found on the medium that would create an invalid filename, it is  
 21762       implementation-dependent if the data from the file is stored on the local file system and under  
 21763       what name it is stored. The *pax* utility may choose to ignore these files as long as it produces an  
 21764       error indicating that the file is being ignored.

### 21765       **The cpio File Data**

21766       Following **c\_name**, there are **c\_filesize** octets of data. Interpretation of such data occurs in a  
 21767       manner dependent on the file. If **c\_filesize** is zero, no data will be contained in **c\_filedata**.

21768       When restoring from an archive:

- 21769       • If the user does not have the appropriate privilege to create a file of the specified type, *pax*  
 21770       will ignore the entry and write an error message to standard error.
- 21771       • Only regular files have data to be restored. Presuming a regular file meets any selection  
 21772       criteria that might be imposed on the format-reading utility by the user, such data will be  
 21773       restored.
- 21774       • If a user does not have appropriate privilege to set a particular mode flag, the flag will be  
 21775       ignored. Some of the mode flags in the archive format are not mentioned elsewhere in this  
 21776       specification. If the implementation does not support those flags, they may be ignored.

### 21777       **The cpio Special Entries**

21778       FIFO special files, directories and the trailer are recorded with **c\_filesize** equal to zero. For other  
 21779       special files, **c\_filesize** is unspecified by this specification. The header for the next file entry in  
 21780       the archive will be written directly after the last octet of the file entry preceding it. A header  
 21781       denoting the filename TRAILER!!! indicates the end of the archive; the contents of octets in the  
 21782       last block of the archive following such a header are undefined.

### Extended tar Format

An extended *tar* archive file contains a series of blocks. Each block will be a fixed-size block of 512 octets (see below). Each file archived is represented by a header block that describes the file, followed by zero or more blocks that give the contents of the file. At the end of the archive file there will be two blocks filled with binary zeros, interpreted as an end-of-archive indicator.

The blocks may be grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the *pax -b* option) may be written with a single write operation. On magnetic tape, the result of this write will be a single tape record. The last group of blocks always will be at the full size, so blocks after the two zero blocks may contain undefined data.

The header block will be structured as shown in the following table. All lengths and offsets are in decimal.

| Field Name      | Octet Offset | Length (in Octets) |
|-----------------|--------------|--------------------|
| <i>name</i>     | 0            | 100                |
| <i>mode</i>     | 100          | 8                  |
| <i>uid</i>      | 108          | 8                  |
| <i>gid</i>      | 116          | 8                  |
| <i>size</i>     | 124          | 12                 |
| <i>mtime</i>    | 136          | 12                 |
| <i>chksum</i>   | 148          | 8                  |
| <i>typeflag</i> | 156          | 1                  |
| <i>linkname</i> | 157          | 100                |
| <i>magic</i>    | 257          | 6                  |
| <i>version</i>  | 263          | 2                  |
| <i>uname</i>    | 265          | 32                 |
| <i>gname</i>    | 297          | 32                 |
| <i>devmajor</i> | 329          | 8                  |
| <i>devminor</i> | 337          | 8                  |
| <i>prefix</i>   | 345          | 155                |

**Table 3-13** Extended *tar* Header Block

All characters in the header block are represented in the coded character set of the ISO/IEC 646:1991 standard. For maximum portability between implementations, names should be selected from characters represented by the portable filename character set as octets with the most significant bit zero. If an implementation supports the use of characters outside the portable filename character set in names for files, users and groups, one or more implementation-dependent encodings of these characters will be provided for interchange purposes. However, the *pax* utility will never create filenames on the local system that cannot be accessed via the procedures described previously in this specification. If a filename is found on the medium that would create an invalid filename, it is implementation-dependent if the data from the file is stored on the file hierarchy and under what name it is stored. The *pax* utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

Each field within the header block is contiguous; that is, there is no padding used. Each character on the archive medium is stored contiguously.

The fields **magic**, **uname** and **gname** are character strings each terminated by a NUL character. The fields **name**, **linkname** and **prefix** are NUL-terminated character strings except when all characters in the array contain non-NUL characters including the last character. The **version** field is two octets containing the characters **00** (zero-zero). The *typeflag* contains a single

21831 character. All other fields are leading zero-filled octal numbers using digits from the  
 21832 ISO/IEC 646:1991 standard IRV. Each numeric field is terminated by one or more space or NUL  
 21833 characters.

21834 The **name** and the **prefix** fields produce the pathname of the file. The hierarchical relationship of  
 21835 the file can be retained by specifying the pathname as a path prefix, and a slash character and  
 21836 filename as the suffix. A new pathname is formed, if *prefix* is not an empty string (its first  
 21837 character is not NUL), by concatenating *prefix* (up to the first NUL character), a slash character  
 21838 and *name*; otherwise, *name* is used alone. In either case, *name* is terminated at the first NUL  
 21839 character. If *prefix* begins with a NUL character, it will be ignored. In this manner, pathnames of  
 21840 at most 256 characters can be supported. If a pathname does not fit in the space provided, *pax*  
 21841 will notify the user of the error, and will not store any part of the file header or data on the  
 21842 medium.

21843 The **linkname** field, described below, does not use the *prefix* to produce a pathname. As such, a  
 21844 *linkname* is limited to 100 characters. If the name does not fit in the space provided, *pax* will  
 21845 notify the user of the error, and will not attempt to store the link on the medium.

21846 The **mode** field provides 12 bits encoded in the ISO/IEC 646:1991 standard octal digit  
 21847 representation. The encoded bits represent the following values:

|       |               |                          |
|-------|---------------|--------------------------|
| 21848 | <b>04 000</b> | Set UID on execution.    |
| 21849 | <b>02 000</b> | Set GID on execution.    |
| 21850 | <b>01 000</b> | Reserved.                |
| 21851 | <b>00 400</b> | Read by owner.           |
| 21852 | <b>00 200</b> | Write by owner.          |
| 21853 | <b>00 100</b> | Execute/search by owner. |
| 21854 | <b>00 040</b> | Read by group.           |
| 21855 | <b>00 020</b> | Write by group.          |
| 21856 | <b>00 010</b> | Execute/search by group. |
| 21857 | <b>00 004</b> | Read by other.           |
| 21858 | <b>00 002</b> | Write by other.          |
| 21859 | <b>00 001</b> | Execute/search by other. |

21860 When appropriate privilege is required to set one of these mode bits, and the user restoring the  
 21861 files from the archive does not have the appropriate privilege, the mode bits for which the user  
 21862 does not have appropriate privilege will be ignored. Some of the mode bits in the archive format  
 21863 are not mentioned elsewhere in this specification. If the implementation does not support those  
 21864 bits, they may be ignored.

21865 The **uid** and **gid** fields are the user and group ID of the owner and group of the file, respectively.

21866 The **size** field is the size of the file in octets. If the **typeflag** field is set to specify a file to be of  
 21867 type 1 (a link) or 2 (reserved for symbolic links), the **size** field will be specified as zero. If the  
 21868 **typeflag** field is set to specify a file of type 5 (directory), the **size** field will be interpreted as  
 21869 described under the definition of that record type. No data blocks will be stored for types 1, 2 or  
 21870 5. If the **typeflag** field is set to 3 (character special file), 4 (block special file), or 6 (FIFO), the  
 21871 meaning of the **size** field is unspecified by this specification, and no data blocks will be stored on  
 21872 the medium. Additionally, for 6, the **size** field is ignored when reading. If the **typeflag** field is  
 21873 set to any other value, the number of blocks written following the header will be  $(\text{size}+511)/512$ ,  
 21874 ignoring any fraction in the result of the division.

21875 The **mtime** field is the modification time of the file at the time it was archived. It is the  
 21876 ISO/IEC 646:1991 standard representation of the octal value of the modification time obtained  
 21877 from the **XSH** specification *stat()* function.

21878 The **chksum** field is the ISO/IEC 646: 1991 standard IRV representation of the octal value of the  
 21879 simple sum of all octets in the header block. Each octet in the header is treated as an unsigned  
 21880 value. These values will be added to an unsigned integer, initialised to zero, the precision of  
 21881 which will be not less than 17 bits. When calculating the checksum, the **chksum** field is treated  
 21882 as if it were all spaces.

21883 The **typeflag** field specifies the type of file archived. If a particular implementation does not  
 21884 recognise the type, or the user does not have appropriate privilege to create that type, the file  
 21885 will be extracted as if it were a regular file if the file type is defined to have a meaning for the  
 21886 **size** field that could cause data blocks to be written on the medium (see the previous description  
 21887 for *size*). If conversion to a regular file occurs, the *pax* utility will produce an error indicating  
 21888 that the conversion took place. All of the **typeflag** fields will be coded in the ISO/IEC 646: 1991  
 21889 standard IRV:

21890 '0' Represents a regular file. For backward compatibility, a **typeflag** value of binary zero  
 21891 ('\0') should be recognised as meaning a regular file when extracting files from the  
 21892 archive. Archives written with this version of the archive file format create regular files  
 21893 with a **typeflag** value of the ISO/IEC 646: 1991 standard IRV 0.

21894 '1' Represents a file linked to another file, of any type, previously archived. Such files are  
 21895 identified by each file having the same device and file serial number. The linked-to  
 21896 name is specified in the **linkname** field with a NUL-character terminator if it is less  
 21897 than 100 octets in length.

21898 '2' Reserved to represent a link to another file, of any type, whose device or file serial  
 21899 number differs. This is provided for systems that support linked files whose device or  
 21900 file serial numbers differ, and should be treated as a type 1 file if this extension does not  
 21901 exist.

21902 '3','4' Represent character special files and block special files respectively. In this case the  
 21903 **devmajor** and **devminor** fields contain information defining the device, the format of  
 21904 which is unspecified by this specification. Implementations may map the device  
 21905 specifications to their own local specification or may ignore the entry.

21906 '5' Specifies a directory or subdirectory. On systems where disk allocation is performed  
 21907 on a directory basis, the **size** field will contain the maximum number of octets (which  
 21908 may be rounded to the nearest disk block allocation unit) that the directory may hold.  
 21909 A **size** field of zero indicates no such limiting. Systems that do not support limiting in  
 21910 this manner should ignore the **size** field.

21911 '6' Specifies a FIFO special file. Note that the archiving of a FIFO file archives the  
 21912 existence of this file and not its contents.

21913 '7' Reserved to represent a file to which an implementation has associated some high-  
 21914 performance attribute. Implementations without such extensions should treat this file  
 21915 as a regular file (type 0).

21916 'A'-'Z' The letters A to Z, inclusive, are reserved for custom implementations. All other values  
 21917 are reserved for specification in future issues.

21918 The **magic** field is the specification that this archive was output in this archive format. If this  
 21919 field contains **ustar** (the five characters from the ISO/IEC 646: 1991 standard IRV shown  
 21920 followed by NUL), the **uname** and **gname** fields will contain the ISO/IEC 646: 1991 standard IRV  
 21921 representation of the owner and group of the file respectively (truncated to fit, if necessary).  
 21922 When the file is restored by a privileged, protection-preserving version of the utility, the  
 21923 password and group files will be scanned for these names. If found, the user and group IDs  
 21924 contained within these files will be used rather than the values contained within the **uid** and **gid**



21925 fields.

## 21926 EXIT STATUS

21927 The following exit values are returned:

21928 0 All files were processed successfully.

21929 >0 An error occurred.

## 21930 CONSEQUENCES OF ERRORS

21931 If *pax* cannot create a file or a link when reading an archive or cannot find a file when writing an  
 21932 archive, or cannot preserve the user ID, group ID or file mode when the **-p** option is specified, a  
 21933 diagnostic message will be written to standard error and a non-zero exit status will be returned,  
 21934 but processing will continue. In the case where *pax* cannot create a link to a file, *pax* will not, by  
 21935 default, create a second copy of the file.

21936 If the extraction of a file from an archive is prematurely terminated by a signal or error, *pax* may  
 21937 have only partially extracted the file or (if the **-n** option was not specified) may have extracted a  
 21938 file of the same name as that specified by the user, but which is not the file the user wanted.  
 21939 Additionally, the file modes of extracted directories may have additional bits from the S\_IRWXU  
 21940 mask set as well as incorrect modification and access times.

## 21941 APPLICATION USAGE

21942 The **-p** (privileges) option was invented to reconcile differences between historical *tar* and *cpio*  
 21943 implementations. In particular, the two utilities use **-m** in diametrically opposed ways. The **-p**  
 21944 option also provides a consistent means of extending the ways in which future file attributes can  
 21945 be addressed, such as for enhanced security systems or high-performance files. Although it may  
 21946 seem complex, there are really two modes that will be most commonly used:

21947 **-p e** “Preserve everything”. This would be used by the historical superuser, someone with  
 21948 all the appropriate privileges, to preserve all aspects of the files as they are recorded in  
 21949 the archive. The **e** flag is the sum of **o** and **p**, and other implementation-dependent  
 21950 attributes.

21951 **-p p** “Preserve” the file mode bits. This would be used by the user with regular privileges  
 21952 who wished to preserve aspects of the file other than the ownership. The file times are  
 21953 preserved by default, but two other flags are offered to disable these and use the time  
 21954 of extraction.

21955 The one pathname per line format of standard input precludes pathnames containing newline  
 21956 characters. Although such pathnames violate the portable filename guidelines, they may exist  
 21957 and their presence may inhibit usage of *pax* within shell scripts. This problem is inherited from  
 21958 historical archive programs. The problem can be avoided by listing filename arguments on the  
 21959 command line instead of on standard input.

21960 It is almost certain that appropriate privileges will be required for *pax* to accomplish parts of this  
 21961 specification. Specifically, creating files of type block special or character special, restoring file  
 21962 access times unless the files are owned by the user (the **-t** option) or preserving file owner,  
 21963 group, and mode (the **-p** option) will all probably require appropriate privileges.

21964 In read mode, implementations are permitted to overwrite files when the archive has multiple  
 21965 members with the same name. This may fail if permissions on the first version of the file do not  
 21966 permit it to be overwritten.

21967 The *cpio* and *tar* formats can only support files up to 8 gigabytes in size.

21968 The *pax* utility is not able to handle arbitrary file sizes. There is currently a proposal in ballot in  
 21969 the IEEE PASC Shell and Utilities Working Group to address this issue.

21970 **EXAMPLES**

21971 The following command:

21972 `pax -w -f /dev/rmt/lm .`21973 copies the contents of the current directory to tape drive 1, medium density (assuming historical  
21974 System V device naming procedures. The historical BSD device name would be /dev/rmt9).

21975 The following commands:

21976 `mkdir newdir`21977 `pax -rw olddir newdir`21978 copy the *olddir* directory hierarchy to *newdir*.21979 `pax -r -s ',^//*usr//*,,' -f a.pax`21980 reads the archive **a.pax**, with all files rooted in **/usr** in the archive extracted relative to the current  
21981 directory.21982 **FUTURE DIRECTIONS**21983 It is expected that future versions of the ISO/IEC 9945-2: 1993 standard will offer additional file  
21984 formats and the **-o** option will be used by future issues to specify such features as international  
21985 filename and file codeset translations, security, accounting, and so on, related to each additional  
21986 format.21987 EX The *pax* utility is not able to handle arbitrary file sizes. There is currently a proposal in ballot in  
21988 the IEEE PASC Shell and Utilities Working Group to address this issue.21989 **SEE ALSO**

21990 None.

21991 **CHANGE HISTORY**

21992 First released in Issue 4.

21993 **Issue 5**21994 A note is added to the APPLICATION USAGE indicating that the *cpio* and *tar* formats can only  
21995 support files up to 8 gigabytes in size.

21996 **NAME**21997           pcat — expand and concatenate files (**LEGACY**)21998 **SYNOPSIS**21999 EX       pcat *file...*22000 **DESCRIPTION**

22001       The *pcat* utility unpacks files in the format used by *pack* and writes the unpacked form to  
 22002       standard output. For each *file* operand, a file named *file.z* (or just *file*, if *file* ends in *.z*) is  
 22003       unpacked.

22004       A file is not written in its unpacked form if:

- 22005           • The file cannot be opened.
- 22006           • The file does not appear to be the output of *pack*.

22007 **OPTIONS**

22008       None.

22009 **OPERANDS**

22010       The following operand is supported:

22011       *file*       A pathname of a file to be *pcated*; *file* can include or omit the *.z* suffix.22012 **STDIN**

22013       Not used.

22014 **INPUT FILES**22015       The input files are regular files in the format produced by the *pack* utility.22016 **ENVIRONMENT VARIABLES**22017       The following environment variables may affect the execution of *pcat*:

22018       *LANG*       Provide a default value for the internationalisation variables that are unset or null. If  
 22019       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 22020       default locale will be used. If any of the internationalisation variables contains an  
 22021       invalid setting, the utility will behave as if none of the variables had been defined.

22022       *LC\_ALL*

22023           If set to a non-empty string value, override the values of all the other  
 22024       internationalisation variables.

22025       *LC\_CTYPE*

22026           Determine the locale for the interpretation of sequences of bytes of text data as  
 22027       characters (for example, single- as opposed to multi-byte characters in arguments).

22028       *LC\_MESSAGES*

22029           Determine the locale that should be used to affect the format and contents of diagnostic  
 22030       messages written to standard error.

22031       *NLSPATH*22032           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.22033 **ASYNCHRONOUS EVENTS**

22034       Default.

22035 **STDOUT**22036       The standard output is the concatenation of the unpacked files identified by the *file* operands.

22037 **STDERR**

22038           Used only for diagnostic messages.

22039 **OUTPUT FILES**

22040           None.

22041 **EXTENDED DESCRIPTION**

22042           None.

22043 **EXIT STATUS**

22044           The following exit values are returned:

22045           0   Successful completion.

22046           &gt;0  An error occurred.

22047 **CONSEQUENCES OF ERRORS**

22048           Default.

22049 **APPLICATION USAGE**22050           The *pcat* utility does for packed files what *cat* does for ordinary files, except that *pcat* cannot be  
22051           used as a filter.22052           Applications should migrate to the *zcat* utility.22053 **EXAMPLES**22054           To view a packed file named *file.z* use:22055           `pcat file.z`

22056           or:

22057           `pcat file`22058           To make an unpacked copy, called **abc**, of a packed file named *file.z* (without destroying *file.z*)  
22059           use:22060           `pcat file >abc`22061 **FUTURE DIRECTIONS**

22062           None.

22063 **SEE ALSO**22064           *pack*, *unpack*, *zcat*.22065 **CHANGE HISTORY**

22066           First released in Issue 2.

22067 **Issue 4**

22068           Format reorganised.

22069           Split into a separate description.

22070           Marked TO BE WITHDRAWN.

22071           Internationalised environment variable support made optional.

22072 **Issue 4, Version 2**22073           An assertion formerly in the DESCRIPTION section, that a file is not written if the filename has  
22074           more than {NAME\_MAX}-2 bytes, is deleted.22075 **Issue 5**

22076           Marked LEGACY.

22077 **NAME**22078 `pg` — file perusal filter for soft-copy terminals (**LEGACY**)22079 **SYNOPSIS**22080 `EX` `pg[-number][-cefn][-p string][+linenumber][+/re/][file...]`22081 **DESCRIPTION**

22082 The `pg` utility is a filter that allows the examination of the named *file* or files one screenful at a  
 22083 time on a soft-copy terminal. Each screenful is followed by a prompt. If the user types a  
 22084 carriage-return character, another page is displayed; other possibilities are enumerated in the  
 22085 EXTENDED DESCRIPTION section.

22086 If the standard output is not a terminal, `pg` ignores all options and writes the input files to  
 22087 standard output, like the `cat` utility, except that a header is written before each file (if there is  
 22088 more than one).

22089 **OPTIONS**

22090 The `pg` utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 22091 that the `-number` option takes a multi-digit value and the `+linenumber` and `+/re/` options take a  
 22092 leading plus sign instead of minus. The following options are supported when standard input is  
 22093 a terminal:

22094 `-number`

22095 Change window size. The *number* argument is a positive integer specifying the size (in  
 22096 lines) of the window that `pg` is to use instead of the default. (The default size is  
 22097 determined by the `LINES` environment variable. If `LINES` is unset or null, a terminal-  
 22098 specific default size is used; the `TERM` environment variable may affect the  
 22099 determination of the terminal-specific default size. The default is typically one less  
 22100 than the number of lines in the terminal window.)

22101 `-p string`

22102 Use *string* as the prompt. If the prompt string contains a `%d`, the first occurrence of `%d`  
 22103 in the prompt will be replaced by the current page number when the prompt is issued.  
 22104 The default prompt string is `:"`.

22105 `-c` Home the cursor and clear the screen before displaying each page. This option is  
 22106 ignored if the terminal does not support this.

22107 `-e` Do not pause at the end of each file; see the EXTENDED DESCRIPTION section.

22108 `-f` Inhibit line splitting. Normally, `pg` splits lines longer than the screen width, but some  
 22109 sequences of characters in the text being displayed (for example, escape sequences for  
 22110 underlining) generate undesirable results.

22111 `UN` `-n` Cause an automatic end-of-command as soon as a command letter is entered.  
 22112 Normally, commands must be terminated by a newline character. The `-n` option is not  
 22113 supported on certain block-mode terminals. If `pg` is being used on such terminal, the `-n`  
 22114 option will be silently ignored and the terminating newline character will be required  
 22115 to complete a command.

22116 `-s` Write all messages and prompts in standout mode (usually inverse video) if the  
 22117 terminal supports this.

22118 `+linenumber`22119 Start up at *linenumber*.22120 `+/re/` Start up at the first line matching the basic regular expression *re*.

**22121 OPERANDS**

22122 The following operands are supported:

22123 *file* A pathname of a text file to be displayed. If no *file* is given, or if it is –, the standard  
22124 input is read.

**22125 STDIN**

22126 The standard input is a text file that is used if no *file* operand is given, or if it is –.

**22127 INPUT FILES**

22128 The input file named by the *file* operand is a text file.

**22129 ENVIRONMENT VARIABLES**

22130 The following environment variables affect the execution of *pg*:

**22131 COLUMNS**

22132 Determine the horizontal screen size. If unset or null, use the value of *TERM*, the  
22133 window size, baud rate, or some combination of these, to indicate the terminal type for  
22134 the screen size calculation.

22135 *LINES* Determine the number of lines to be displayed on the screen. If unset or null, use the  
22136 value of *TERM*, the window size, baud rate, or some combination of these, to indicate  
22137 the terminal type for the screen size calculation.

22138 *SHELL* Determine the name of the command interpreter executed for a !command.

22139 *TERM* Determine terminal attributes. Optionally attempt to search a system-dependent  
22140 database, keyed on the value of the *TERM* environment variable. If no information is  
22141 available, a terminal incapable of cursor-addressable movement is assumed.

22142 The following environment variables may affect the execution of *pg*:

22143 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
22144 *LANG* is unset or null, the corresponding value from the implementation-dependent  
22145 default locale will be used. If any of the internationalisation variables contains an  
22146 invalid setting, the utility will behave as if none of the variables had been defined.

**22147 LC\_ALL**

22148 If set to a non-empty string value, override the values of all the other  
22149 internationalisation variables.

**22150 LC\_COLLATE**

22151 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
22152 character collating elements within regular expressions.

**22153 LC\_CTYPE**

22154 Determine the locale for the interpretation of sequences of bytes of text data as  
22155 characters (for example, single- as opposed to multi-byte characters in arguments and  
22156 input files) and the behaviour of character classes within regular expressions.

**22157 LC\_MESSAGES**

22158 Determine the locale that should be used to affect the format and contents of diagnostic  
22159 messages written to standard error, and informative messages written to standard  
22160 output.

**22161 NLSPATH**

22162 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

**22163 ASYNCHRONOUS EVENTS**

22164 During the writing of characters to the standard output, *pg* catches SIGINT and SIGQUIT,

22165 returning the user to the prompt. When waiting at the prompt, *pg* takes the default action on  
 22166 SIGINT and SIGQUIT. The default actions are taken for all other signals.

#### 22167 **STDOUT**

22168 The standard output is a display of the input files, with prompts interspersed. When standard  
 22169 output is not associated with a terminal, it consists of the contents of the input files, separated by  
 22170 the following lines, if there is more than one input file:

22171 "::::::::::::\n%s\n::::::::::::\n", *<pathname>*

#### 22172 **STDERR**

22173 Used only for diagnostic messages.

#### 22174 **OUTPUT FILES**

22175 None.

#### 22176 **EXTENDED DESCRIPTION**

22177 The responses that can be typed when *pg* pauses can be divided into three categories: those  
 22178 causing further perusal, those that search and those that modify the perusal environment.

22179 Commands that cause further perusal normally take a preceding address; an optionally signed  
 22180 number indicating the point from which further text should be displayed. This address is  
 22181 interpreted in either pages or lines depending on the command. A signed address specifies a  
 22182 point relative to the current page or line, and an unsigned address specifies an address relative  
 22183 to the beginning of the file. Each command has a default address that is used if none is  
 22184 provided.

22185 The perusal commands and their defaults are as follows:

22186 (+1)<newline> or <space>

22187 Display one page. The address is specified in number of pages.

22188 (+1)l With a relative address, simulate scrolling the screen, forward or backward, by the  
 22189 number of lines specified. With an absolute address, print a screenful beginning at the  
 22190 specified line.

22191 (+1)d or <control>-D

22192 Simulate scrolling half a screen forward or backward.

22193 The following perusal commands take no address:

22194 . or <control>-L

22195 Redisplay the current page of text.

22196 \$ Display the last windowful in the file.

22197 The following commands are available for searching for text patterns in the text. Basic regular  
 22198 expression syntax is used in patterns. The *res* must always be terminated by a newline character,  
 22199 even if the *-n* option is specified.

22200 *i/re/* Search forwards for the *i*th occurrence of *re* (default *i*=1). Searching begins immediately  
 22201 after the current page and continues to the end of the current file, without wrap-  
 22202 around.

22203 *^re*

22204 *^re?* Search backwards for the *i*th occurrence of *re* (default *i*=1). Searching begins  
 22205 immediately before the current page and continues to the beginning of the current file,  
 22206 without wrap-around.

22207 After searching, *pg* will normally display the line found at the top of the screen. This can be  
 22208 modified by appending **m** or **b** to the search command to leave the line found in the middle or at  
 22209 the bottom of the window from now on. The suffix **t** can be used to restore the original  
 22210 situation.

22211 If the standard output is not a terminal device, *pg* always exits when it reaches end-of-file on the  
 22212 last file in its argument list. Otherwise, for all files but the last, *pg* prompts with an indication  
 22213 that it has reached the end-of-file, along with the name of the next file; when **-e** is used, the end-  
 22214 of-file prompt is bypassed and the prompt indicates the name of the next file. For the last file  
 22215 specified, or for the standard input if no file is specified, *pg* prompts indicating end-of-file (if **-e**  
 22216 is not used), and accepts additional commands. If the next command specifies forward  
 22217 scrolling, *pg* exits. If the **-e** option is specified, *pg* exits immediately after writing the last line of  
 22218 the last file.

22219 The user of *pg* can modify the environment of perusal with the following commands:

22220 **in** Begin perusing the *i*th next file in the command line. The *i* is an unsigned number;  
 22221 default value is 1.

22222 **ip** Begin perusing the *i*th previous file in the command line. The *i* is an unsigned number;  
 22223 default is 1.

22224 **iw** Display another window of text. If *i* is present, set the window size to *i*.

22225 **s filename**  
 22226 Save the input in the file named *filename*. Only the current file being perused is saved.  
 22227 The blank-character-separation between the **s** and *filename* is optional. This command  
 22228 must always be terminated by a newline character, even if the **-n** option is specified.

22229 **h** Help by displaying an abbreviated summary of available commands.

22230 **q** or **Q** Quit *pg*.

22231 **!command**  
 22232 Pass the argument *command* to the command interpreter, whose name is taken from the  
 22233 *SHELL* environment variable. If *SHELL* is unset or null, *sh* is used as the default  
 22234 command interpreter. This command must always be terminated by a newline  
 22235 character, even if the **-n** option is specified.

22236 At any time when output is being sent to the terminal, receipt of a quit or interrupt signal causes  
 22237 *pg* to stop sending output and to display the prompt. The user may then enter one of the above  
 22238 commands in the normal manner. Unfortunately, some output is lost when this is done, due to  
 22239 the fact that any characters waiting in the terminal's output queue are flushed when the signal  
 22240 occurs.

## 22241 EXIT STATUS

22242 The following exit values are returned:

22243 0 Successful completion.

22244 >0 An error occurred.

## 22245 CONSEQUENCES OF ERRORS

22246 Default.

## 22247 APPLICATION USAGE

22248 This utility is different from previous paginators in that it allows the user to back up and review  
 22249 something that has already passed. The method for doing this is explained in the EXTENDED  
 22250 DESCRIPTION section.



- 22251 While waiting for terminal input, *pg* responds to SIGINT and SIGQUIT signals by terminating  
22252 execution. Between prompts, however, these signals interrupt *pg*'s current task and place the  
22253 user in prompt mode. These signals should be used with caution when input is being read from  
22254 a pipe, since an interrupt is likely to terminate the other commands in the pipeline.
- 22255 Use the \$ command with caution when the input is a pipe.
- 22256 If terminal tabs are not set every eight column positions, undesirable results may occur.
- 22257 When *pg* is used as a filter with another command that changes the terminal I/O options,  
22258 terminal settings might not be restored correctly.
- 22259 Applications should migrate to the *more* utility.
- 22260 **EXAMPLES**
- 22261 None.
- 22262 **FUTURE DIRECTIONS**
- 22263 None.
- 22264 **SEE ALSO**
- 22265 *more*, the XBD specification, **Section 7.3, Basic Regular Expressions**, the XBD specification,  
22266 **Chapter 9, General Terminal Interface**.
- 22267 **CHANGE HISTORY**
- 22268 First released in Issue 2.
- 22269 **Issue 4**
- 22270 Format reorganised.
- 22271 Exceptions to Utility Syntax Guidelines conformance noted.
- 22272 Internationalised environment variable and regular expression support made optional.
- 22273 Marked TO BE WITHDRAWN.
- 22274 **Issue 5**
- 22275 Marked LEGACY.

## 22276 NAME

22277 pr — print files

## 22278 SYNOPSIS

22279 pr [+page][-column][-adFmrt][-e[*char*][*gap*]][-h header][-i[*char*][*gap*]]  
 22280 EX [-l *lines*][-n[*char*][*width*]][-o *offset*][-s[*char*]][-w *width*][-fp]  
 22281 [*file...*]

## 22282 DESCRIPTION

22283 The *pr* utility is a printing and pagination filter. If multiple input files are specified, each is read,  
 22284 formatted, and written to standard output. By default, the input is separated into 66-line pages,  
 22285 each with:

- 22286 • a 5-line header that includes the page number, date, time and the pathname of the file
- 22287 • a 5-line trailer consisting of blank lines.

22288 If standard output is associated with a terminal, diagnostic messages will be deferred until the *pr*  
 22289 utility has completed processing.

22290 When options specifying multi-column output are specified, output text columns will be of  
 22291 equal width; input lines that do not fit into a text column will be truncated. By default, text  
 22292 columns are separated with at least one blank character.

## 22293 OPTIONS

22294 The *pr* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 22295 that: the *page* option has a "+" delimiter; *page* and *column* can be multi-digit numbers; some of  
 22296 the option-arguments are optional; and some of the option-arguments cannot be specified as  
 22297 separate arguments from the preceding option letter. In particular, the *-s* option does not allow  
 22298 the option letter to be separated from its argument, and the options *-e*, *-i* and *-n* require that  
 22299 both arguments, if present, not be separated from the option letter.

22300 The following options are supported. In the following option descriptions, *column*, *lines*, *offset*,  
 22301 *page* and *width* are positive decimal integers; *gap* is a non-negative decimal integer.

22302 *+page* Begin output at page number *page* of the formatted input.

22303 *-column* Produce multi-column output that is arranged in *column* columns (default is 1) and is  
 22304 written down each column in the order in which the text is received from the input file.  
 22305 This option should not be used with *-m*. The options *-e* and *-i* will be assumed for  
 22306 multiple text-column output. Whether or not text columns are produced with identical  
 22307 vertical lengths is unspecified, but a text column will never exceed the length of the  
 22308 page (see the *-l* option). When used with *-t*, use the minimum number of lines to write  
 22309 the output.

22310 *-a* Modify the effect of the *-column* option so that the columns are filled across the page in  
 22311 a round-robin order (for example, when *column* is 2, the first input line heads column 1,  
 22312 the second heads column 2, the third is the second line in column 1, and so on).

22313 *-d* Produce output that is double-spaced; append an extra newline character following  
 22314 every newline character found in the input.

22315 *-e[*char*][*gap*]*

22316 Expand each input tab character to the next greater column position specified by the  
 22317 formula  $n * gap + 1$ , where *n* is an integer > 0. If *gap* is zero or is omitted, it defaults to 8.  
 22318 All tab characters in the input will be expanded into the appropriate number of space  
 22319 characters. If any non-digit character, *char*, is specified, it will be used as the input tab  
 22320 character.

|          |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22321 EX | <b>-f</b>                | Use a form-feed character for new pages, instead of the default behaviour that uses a sequence of newline characters. Pause before beginning the first page if the standard output is associated with a terminal.                                                                                                                                                                                                |
| 22322    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22323    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22324    | <b>-F</b>                | Use a form-feed character for new pages, instead of the default behaviour that uses a sequence of newline characters.                                                                                                                                                                                                                                                                                            |
| 22325    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22326    | <b>-h header</b>         |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22327    |                          | Use the string <i>header</i> to replace the contents of the <i>file</i> operand in the page header.                                                                                                                                                                                                                                                                                                              |
| 22328    | <b>-i [char] [gap]</b>   |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22329    |                          | In output, replace multiple space characters with tab characters wherever two or more adjacent space characters reach column positions <i>gap</i> +1, 2* <i>gap</i> +1, 3* <i>gap</i> +1, and so on. If <i>gap</i> is zero or is omitted, default tab settings at every eighth column position are assumed. If any non-digit character, <i>char</i> , is specified, it will be used as the output tab character. |
| 22330    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22331    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22332    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22333    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22334    | <b>-l lines</b>          | Override the 66-line default and reset the page length to <i>lines</i> . If <i>lines</i> is not greater than the sum of both the header and trailer depths (in lines), the <i>pr</i> utility will suppress both the header and trailer, as if the <b>-t</b> option were in effect.                                                                                                                               |
| 22335    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22336    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22337    | <b>-m</b>                | Merge files. Standard output will be formatted so the <i>pr</i> utility writes one line from each file specified by a <i>file</i> operand, side by side into text columns of equal fixed widths, in terms of the number of column positions. Implementations support merging of at least nine <i>file</i> operands.                                                                                              |
| 22338    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22339    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22340    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22341    | <b>-n [char] [width]</b> |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22342    |                          | Provide <i>width</i> -digit line numbering (default for <i>width</i> is 5). The number will occupy the first <i>width</i> column positions of each text column of default output or each line of <b>-m</b> output. If <i>char</i> (any non-digit character) is given, it will be appended to the line number to separate it from whatever follows (default for <i>char</i> is a tab character).                  |
| 22343    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22344    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22345    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22346    | <b>-o offset</b>         |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22347    |                          | Each line of output will be preceded by offset <space> <i>s</i> . If the <b>-o</b> option is not specified, the default offset is zero. The space taken will be in addition to the output line width (see <b>-w</b> option below).                                                                                                                                                                               |
| 22348    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22349    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22350 EX | <b>-p</b>                | Pause before beginning each page if the standard output is directed to a terminal ( <i>pr</i> will write an alert character to standard error and wait for a carriage-return character to be read on <i>/dev/tty</i> ).                                                                                                                                                                                          |
| 22351    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22352    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22353    | <b>-r</b>                | Write no diagnostic reports on failure to open files.                                                                                                                                                                                                                                                                                                                                                            |
| 22354    | <b>-s [char]</b>         |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22355    |                          | Separate text columns by the single character <i>char</i> instead of by the appropriate number of space characters (default for <i>char</i> is the tab character).                                                                                                                                                                                                                                               |
| 22356    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22357    | <b>-t</b>                | Write neither the five-line identifying header nor the five-line trailer usually supplied for each page. Quit writing after the last line of each file without spacing to the end of the page.                                                                                                                                                                                                                   |
| 22358    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22359    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22360    | <b>-w width</b>          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22361    |                          | Set the width of the line to <i>width</i> column positions for multiple text-column output only. If the <b>-w</b> option is not specified and the <b>-s</b> option is not specified, the default width is 72. If the <b>-w</b> option is not specified and the <b>-s</b> option is specified, the default width is 512.                                                                                          |
| 22362    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22363    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22364    |                          |                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22365    |                          | For single column output, input lines will not be truncated.                                                                                                                                                                                                                                                                                                                                                     |

22366 **OPERANDS**

22367 The following operand is supported:

22368 *file* A pathname of a file to be written. If no *file* operands are specified, or if a *file* operand is  
 22369 "-", the standard input will be used.

22370 **STDIN**

22371 The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".  
 22372 See the INPUT FILES section.

22373 **INPUT FILES**

22374 The input files must be text files.

22375 EX The file `/dev/tty` is used to read responses required by the `-p` option.22376 **ENVIRONMENT VARIABLES**22377 The following environment variables affect the execution of *pr*:

22378 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 22379 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 22380 default locale will be used. If any of the internationalisation variables contains an  
 22381 invalid setting, the utility will behave as if none of the variables had been defined.

22382 *LC\_ALL*

22383 If set to a non-empty string value, override the values of all the other  
 22384 internationalisation variables.

22385 *LC\_CTYPE*

22386 Determine the locale for the interpretation of sequences of bytes of text data as  
 22387 characters (for example, single- versus multi-byte characters in arguments and input  
 22388 files) and which characters are defined as printable (character class **print**). Non-  
 22389 printable characters still will be written to standard output, but are not counted for the  
 22390 purpose for column-width and line-length calculations.

22391 *LC\_MESSAGES*

22392 Determine the locale that should be used to affect the format and contents of diagnostic  
 22393 messages written to standard error.

22394 *LC\_TIME*

22395 Determine the format of the date and time for use in writing header lines.

22396 EX *NLSPATH*22397 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.22398 *TZ* Determine the timezone for use in writing header lines.22399 **ASYNCHRONOUS EVENTS**

22400 If *pr* receives an interrupt while writing to a terminal, it will flush all accumulated error  
 22401 messages to the screen before terminating.

22402 **STDOUT**

22403 The *pr* utility output will be a paginated version of the original file (or files). This pagination  
 22404 will be accomplished using either form-feed characters or a sequence of newline characters, as  
 22405 EX controlled by the `-F` or `-f` option. Page headers will be generated unless the `-t` option is  
 22406 specified. The page headers will be of the form:

22407 "\n\n%s %s Page %d\n\n\n", &lt;output of date&gt;, &lt;file&gt;, &lt;page number&gt;

22408 In the POSIX locale, the <output of date> field, representing the date and time of last modification  
 22409 of the input file (or the current date and time if the input file is standard input), is equivalent to

- 22410 the output of the following command as it would appear if executed at the given time:
- 22411 `date "+%b %e %H:%M %Y"`
- 22412 without the trailing newline character, if the page being written is from standard input. If the  
 22413 page being written is not from standard input, in the POSIX locale, the same format will be used,  
 22414 but the time used will be the modification time of the file corresponding to *file* instead of the  
 22415 current time. When the LC\_TIME locale category is not set to the POSIX locale, a different  
 22416 format and order of presentation of this field may be used.
- 22417 If the standard input is used instead of a *file* operand, the `<file>` field will be replaced by a null  
 22418 string.
- 22419 If the `-h` option is specified, the `<file>` field will be replaced by the *header* argument.
- 22420 **STDERR**
- 22421 EX Used for diagnostic messages and for alerting the terminal when `-p` is specified.
- 22422 **OUTPUT FILES**
- 22423 None.
- 22424 **EXTENDED DESCRIPTION**
- 22425 None.
- 22426 **EXIT STATUS**
- 22427 The following exit values are returned:
- 22428 0 All files were written successfully.
- 22429 >0 An error occurred.
- 22430 **CONSEQUENCES OF ERRORS**
- 22431 Default.
- 22432 **APPLICATION USAGE**
- 22433 None.
- 22434 **EXAMPLES**
- 22435 1. Print a numbered list of all files in the current directory:
- 22436 `ls -a | pr -n -h "Files in $(pwd)."`
- 22437 2. Print **file1** and **file2** as a double-spaced, three-column listing headed by “file list”:
- 22438 `pr -3d -h "file list" file1 file2`
- 22439 3. Write **file1** on **file2**, expanding tabs to columns 10, 19, 28, ...:
- 22440 `pr -e9 -t <file1 >file2`
- 22441 **FUTURE DIRECTIONS**
- 22442 It is possible that a new interface that conforms to the Utility Syntax Guidelines will be  
 22443 introduced.
- 22444 **SEE ALSO**
- 22445 *expand, lp.*

22446 **CHANGE HISTORY**

22447 First released in Issue 2.

22448 **Issue 4**

22449 Aligned with the ISO/IEC 9945-2: 1993 standard.

22450 **NAME**

22451           printf — write formatted output

22452 **SYNOPSIS**22453           printf *format*[*argument...*]22454 **DESCRIPTION**

22455           The *printf* utility will write formatted operands to the standard output. The *argument* operands  
 22456           will be formatted under control of the *format* operand.

22457 **OPTIONS**

22458           None.

22459 **OPERANDS**

22460           The following operands are supported:

22461           *format*   A string describing the format to use to write the remaining operands; see the  
 22462           EXTENDED DESCRIPTION section.

22463           *argument*

22464           The strings to be written to standard output, under the control of *format*; see the  
 22465           EXTENDED DESCRIPTION section.

22466 **STDIN**

22467           Not used.

22468 **INPUT FILES**

22469           None.

22470 **ENVIRONMENT VARIABLES**22471           The following environment variables affect the execution of *printf*:

22472           *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 22473           *LANG* is unset or null, the corresponding value from the implementation-dependent  
 22474           default locale will be used. If any of the internationalisation variables contains an  
 22475           invalid setting, the utility will behave as if none of the variables had been defined.

22476           *LC\_ALL*

22477           If set to a non-empty string value, override the values of all the other  
 22478           internationalisation variables.

22479           *LC\_CTYPE*

22480           Determine the locale for the interpretation of sequences of bytes of text data as  
 22481           characters (for example, single- as opposed to multi-byte characters in arguments).

22482           *LC\_MESSAGES*

22483           Determine the locale that should be used to affect the format and contents of diagnostic  
 22484           messages written to standard error.

22485           *LC\_NUMERIC*

22486           Determine the locale for numeric formatting. It will affect the format of numbers  
 22487           written using the e, E, f, g and G conversion characters (if supported).

22488 EX       *NLSPATH*22489           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.22490 **ASYNCHRONOUS EVENTS**

22491           Default.

22492 **STDOUT**

22493 See the EXTENDED DESCRIPTION section.

22494 **STDERR**

22495 Used only for diagnostic messages.

22496 **OUTPUT FILES**

22497 None.

22498 **EXTENDED DESCRIPTION**

22499 The *format* operand will be used as the *format* string described in the **XBD** specification,  
 22500 **Chapter 3, File Format Notation** with the following exceptions:

- 22501 • A space character in the format string, in any context other than a flag of a conversion  
 22502 specification, will be treated as an ordinary character that is copied to the output.
- 22503 • A  $\Delta$  character in the format string will be treated as a  $\Delta$  character, not as a space character.
- 22504 • In addition to the escape sequences shown in the **XBD** specification, **Chapter 3, File Format**  
 22505 **Notation** ( $\backslash$ ,  $\backslash$ a,  $\backslash$ b,  $\backslash$ f,  $\backslash$ n,  $\backslash$ r,  $\backslash$ t,  $\backslash$ v),  $\backslash$ ddd, where *ddd* is a one-, two- or three-digit octal  
 22506 number, will be written as a byte with the numeric value specified by the octal number.
- 22507 • The implementation will not precede or follow output from the d or u conversion  
 22508 specifications with blank characters not specified by the *format* operand.
- 22509 • The implementation will not precede output from the o conversion specification with zeros  
 22510 not specified by the *format* operand.
- 22511 • The e, E, f, g and G conversion specifications need not be supported.
- 22512 • An additional conversion character, b, will be supported as follows. The argument will be  
 22513 taken to be a string that may contain backslash-escape sequences. The following backslash-  
 22514 escape sequences will be supported:
  - 22515 — the escape sequences listed in the **XBD** specification, **Chapter 3, File Format Notation**  
 22516 ( $\backslash$ ,  $\backslash$ a,  $\backslash$ b,  $\backslash$ f,  $\backslash$ n,  $\backslash$ r,  $\backslash$ t,  $\backslash$ v), which will be converted to the characters they represent
  - 22517 —  $\backslash$ 0ddd, where *ddd* is a zero-, one-, two- or three-digit octal number that will be converted  
 22518 to a byte with the numeric value specified by the octal number
  - 22519 —  $\backslash$ c, which will not be written and will cause *printf* to ignore any remaining characters in  
 22520 the string operand containing it, any remaining string operands and any additional  
 22521 characters in the *format* operand.
- 22522 The interpretation of a backslash followed by any other sequence of characters is unspecified.
- 22523 Bytes from the converted string will be written until the end of the string or the number of  
 22524 bytes indicated by the precision specification is reached. If the precision is omitted, it will be  
 22525 taken to be infinite, so all bytes up to the end of the converted string will be written.
- 22526 • For each specification that consumes an argument, the next argument operand will be  
 22527 evaluated and converted to the appropriate type for the conversion as specified below.
- 22528 • The *format* operand will be reused as often as necessary to satisfy the argument operands.  
 22529 Any extra c or s conversion specifications will be evaluated as if a null string argument were  
 22530 supplied; other extra conversion specifications will be evaluated as if a zero argument were  
 22531 supplied. If the *format* operand contains no conversion specifications and *argument* operands  
 22532 are present, the results are unspecified.
- 22533 • If a character sequence in the *format* operand begins with a "%" character, but does not form a  
 22534 valid conversion specification, the behaviour is unspecified.



22535 The *argument* operands will be treated as strings if the corresponding conversion character is b, c  
 22536 or s; otherwise, it will be evaluated as a C constant, as described by the ISO C standard, with the  
 22537 following extensions:

- 22538 • A leading plus or minus sign will be allowed.
- 22539 • If the leading character is a single- or double-quote, the value will be the numeric value in the  
 22540 underlying codeset of the character following the single- or double-quote.

22541 If an argument operand cannot be completely converted into an internal value appropriate to  
 22542 the corresponding conversion specification, a diagnostic message will be written to standard  
 22543 error and the utility will not exit with a zero exit status, but will continue processing any  
 22544 remaining operands and will write the value accumulated at the time the error was detected to  
 22545 standard output.

#### 22546 EXIT STATUS

22547 The following exit values are returned:

- 22548 0 Successful completion.
- 22549 >0 An error occurred.

#### 22550 CONSEQUENCES OF ERRORS

22551 Default.

#### 22552 APPLICATION USAGE

22553 The floating-point formatting conversion specifications of *printf()* are not required because all  
 22554 arithmetic in the shell is integer arithmetic. The *awk* utility performs floating-point calculations  
 22555 and provides its own **printf** function. The *bc* utility can perform arbitrary-precision floating-  
 22556 point arithmetic, but does not provide extensive formatting capabilities. (This *printf* utility  
 22557 cannot really be used to format *bc* output; it does not support arbitrary precision.)  
 22558 Implementations are encouraged to support the floating-point conversions as an extension.

22559 Note that this *printf* utility, like the **XSH** specification *printf()* function on which it is based,  
 22560 makes no special provision for dealing with multi-byte characters when using the %c conversion  
 22561 specification or when a precision is specified in a %b or %s conversion specification.  
 22562 Applications should be extremely cautious using either of these features when there are multi-  
 22563 byte characters in the character set.

22564 Field widths and precisions cannot be specified as "\*" since the "\*" can be replaced directly in the  
 22565 *format* operand using shell variable substitution. Implementations can also provide this feature  
 22566 as an extension if they so choose.

22567 Hexadecimal character constants as defined in the ISO C standard are not recognised in the  
 22568 *format* operand because there is no consistent way to detect the end of the constant. Octal  
 22569 character constants are limited to, at most, three octal digits, but hexadecimal character  
 22570 constants are only terminated by a non-hex-digit character. In the ISO C standard, the ##  
 22571 concatenation operator can be used to terminate a constant and follow it with a hexadecimal  
 22572 character to be written. In the shell, concatenation occurs before the *printf* utility has a chance to  
 22573 parse the end of the hexadecimal constant.

22574 The %b conversion specification is not part of the ISO C standard; it has been added here as a  
 22575 portable way to process backslash escapes expanded in string operands as provided by the *echo*  
 22576 utility. See also the APPLICATION USAGE section of *echo* for ways to use *printf* as a  
 22577 replacement for all of the traditional versions of the *echo* utility.

22578 If an argument cannot be parsed correctly for the corresponding conversion specification, the  
 22579 *printf* utility is required to report an error. Thus, overflow and extraneous characters at the end  
 22580 of an argument being used for a numeric conversion are to be reported as errors.

22581 It is not considered an error if an argument operand is not completely used for a *c* or *s*  
 22582 conversion or if a string operand's first or second character is used to get the numeric value of a  
 22583 character.

#### 22584 EXAMPLES

22585 To alert the user and then print and read a series of prompts:

```
22586 printf "\aPlease fill in the following: \nName: "
22587 read name
22588 printf "Phone number: "
22589 read phone
```

22590 To read out a list of right and wrong answers from a file, calculate the percentage correctly, and  
 22591 print them out. The numbers are right-justified and separated by a single tab character. The  
 22592 percentage is written to one decimal place of accuracy:

```
22593 while read right wrong ; do
22594 percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
22595 printf "%2d right\t%2d wrong\t(%s%%)\n" \
22596 $right $wrong $percent
22597 done < database_file
```

22598 The command:

```
22599 printf "%5d%4d\n" 1 21 321 4321 54321
```

22600 produces:

```
22601 1 21
22602 3214321
22603 54321 0
```

22604 Note that the *format* operand is used three times to print all of the given strings and that a 0 was  
 22605 supplied by *printf* to satisfy the last *%4d* conversion specification.

22606 The *printf* utility is required to notify the user when conversion errors are detected while  
 22607 producing numeric output; thus, the following results would be expected on an implementation  
 22608 with 32-bit twos-complement integers when *%d* is specified as the *format* operand:

| Argument    | Standard Output | Diagnostic Output                         |
|-------------|-----------------|-------------------------------------------|
| 5a          | 5               | printf: "5a" not completely converted     |
| 9999999999  | 2147483647      | printf: "9999999999" arithmetic overflow  |
| -9999999999 | -2147483648     | printf: "-9999999999" arithmetic overflow |
| ABC         | 0               | printf: "ABC" expected numeric value      |

22616 The diagnostic message format is not specified, but these examples convey the type of  
 22617 information that should be reported. Note that the value shown on standard output is what  
 22618 would be expected as the return value from the XSH specification function *strtol()*. A similar  
 22619 correspondence exists between *%u* and *strtoul()* and *%e*, *%f* and *%g* (if the implementation  
 22620 supports floating-point conversions) and *strtod()*.

22621 In a locale using the ISO/IEC 646: 1991 standard as the underlying codeset, the command:

22622 `printf "%d\n" 3 +3 -3 \'3 \"+3 "'-3"`

22623 produces:

|       |           |                                                                              |
|-------|-----------|------------------------------------------------------------------------------|
| 22624 | <b>3</b>  | Numeric value of constant 3                                                  |
| 22625 | <b>3</b>  | Numeric value of constant 3                                                  |
| 22626 | <b>-3</b> | Numeric value of constant -3                                                 |
| 22627 | <b>51</b> | Numeric value of the character '3' in the ISO/IEC 646: 1991 standard codeset |
| 22628 | <b>43</b> | Numeric value of the character '+' in the ISO/IEC 646: 1991 standard codeset |
| 22629 | <b>45</b> | Numeric value of the character '-' in the ISO/IEC 646: 1991 standard codeset |
| 22630 |           |                                                                              |

22631 Note that in a locale with multi-byte characters, the value of a character is intended to be the  
 22632 value of the equivalent of the **wchar\_t** representation of the character as described in the **XSH**  
 22633 specification.

#### 22634 **FUTURE DIRECTIONS**

22635 None.

#### 22636 **SEE ALSO**

22637 *awk*, *bc*, *echo*, the **XSH** specification description of *printf()*.

#### 22638 **CHANGE HISTORY**

22639 First released in Issue 4.

## 22640 NAME

22641        `prs` — print an SCCS file (**DEVELOPMENT**)

## 22642 SYNOPSIS

22643 EX        `prs [-a][-d dataspec][-r[SID]] file...`

22644 EX        `prs [ -e | -l ] -c cutoff [-d dataspec] file...`

22645 EX        `prs [ -e | -l ] -r[SID][-d dataspec]file...`

## 22646 DESCRIPTION

22647        The *prs* utility writes to standard output parts or all of an SCCS file in a user-supplied format.

## 22648 OPTIONS

22649        The *prs* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
22650        that the **-r** option has an optional option-argument. This optional option-argument cannot be  
22651        presented as a separate argument. The following options are supported:

22652        **-d *dataspec***

22653            Specify the output data specification. The *dataspec* is a string consisting of SCCS file  
22654            *data keywords* (see **Data Keywords** on page 613) interspersed with optional user-  
22655            supplied text.

22656        **-r[*SID*]**

22657            Specify the SCCS identification string (*SID*) of a delta for which information is desired.  
22658            If no *SID* option-argument is specified, the *SID* of the most recently created delta is  
22659            assumed.

22660        **-e**        Request information for all deltas created earlier than and including the delta  
22661            designated via the **-r** option or the date-time given by the **-c** option.

22662        **-l**        Request information for all deltas created later than and including the delta designated  
22663            via the **-r** option or the date-time given by the **-c** option.

22664        **-c *cutoff***

22665            Indicate the *cutoff* date-time, in the form:

22666            `YY[MM[DD[HH[MM[SS]]]]]`

22667            For the *YY* component, values in the range [69-99] refer to years in the twentieth  
22668            century (1969 to 1999 inclusive); values in the range [00-68] refer to years in the  
22669            twenty-first century (2000 to 2068 inclusive).

22670            No changes (deltas) to the SCCS file that were created after the specified *cutoff* date-  
22671            time are included in the output. Units omitted from the date-time default to their  
22672            maximum possible values; for example, **-c 7502** is equivalent to **-c 750228235959**.

22673        **-a**        Request writing of information for both removed, that is, delta type = *R* (see *rmDEL*) and  
22674            existing, that is, delta type = *D*, deltas. If the **-a** option is not specified, information for  
22675            existing deltas only is provided.

## 22676 OPERANDS

22677        The following operand is supported:

22678        ***file***        A pathname of an existing SCCS file or a directory. If *file* is a directory, *prs* behaves as  
22679            though each file in the directory were specified as a named file, except that non-SCCS  
22680            files (last component of the pathname does not begin with *s.*) and unreadable files are  
22681            silently ignored.

22682            If a single instance *file* is specified as **-**, the standard input is read; each line of the  
22683            standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files

22684 and unreadable files are silently ignored.

22685 **STDIN**

22686 The standard input is a text file used only when the *file* operand is specified as *-*. Each line of the

22687 text file is interpreted as an SCCS pathname.

22688 **INPUT FILES**

22689 Any SCCS files displayed are files of an unspecified format.

22690 **ENVIRONMENT VARIABLES**

22691 The following environment variables affect the execution of *prs*:

22692 *LANG* Provide a default value for the internationalisation variables that are unset or null. If

22693 *LANG* is unset or null, the corresponding value from the implementation-dependent

22694 default locale will be used. If any of the internationalisation variables contains an

22695 invalid setting, the utility will behave as if none of the variables had been defined.

22696 *LC\_ALL*

22697 If set to a non-empty string value, override the values of all the other

22698 internationalisation variables.

22699 *LC\_CTYPE*

22700 Determine the locale for the interpretation of sequences of bytes of text data as

22701 characters (for example, single- as opposed to multi-byte characters in arguments and

22702 input files).

22703 *LC\_MESSAGES*

22704 Determine the locale that should be used to affect the format and contents of diagnostic

22705 messages written to standard error.

22706 *NLSPATH*

22707 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

22708 **ASYNCHRONOUS EVENTS**

22709 Default.

22710 **STDOUT**

22711 The standard output is a text file whose format is dependent on the data keywords specified

22712 with the *-d* option.

22713 **Data Keywords**

22714 Data keywords specify which parts of an SCCS file are to be retrieved and output. All parts of

22715 an SCCS file have an associated data keyword. A data keyword may appear in a *dataspec*

22716 multiple times.

22717 The information written by *prs* consists of: (1) the user-supplied text; and (2) appropriate values

22718 (extracted from the SCCS file) substituted for the recognised data keywords in the order of

22719 appearance in the *dataspec*. The format of a data keyword value is either simple (S), in which

22720 keyword substitution is direct, or multi-line (M).

22721 User-supplied text is any text other than recognised data keywords. A tab character is specified

22722 by *\t* and newline by *\n*. When the *-r* option is not specified, the default *dataspec* is:

22723 :PN: :\n\n

22724 and the following *dataspec* is used for each selected delta:

22725 :Dt: \t:DL: \nMRS: \n:MR: COMMENTS: \n:C: \n

22726

22727

22728

22729

22730

22731

22732

22733

22734

22735

22736

22737

22738

22739

22740

22741

22742

22743

22744

22745

22746

22747

22748

22749

22750

22751

22752

22753

22754

22755

22756

22757

22758

22759

22760

22761

22762

22763

22764

22765

22766

22767

| SCCS File Data Keywords |                                                 |              |                       |        |
|-------------------------|-------------------------------------------------|--------------|-----------------------|--------|
| Keyword                 | Data Item                                       | File Section | Value                 | Format |
| <b>:Dt:</b>             | Delta information                               | Delta Table  | See below*            | S      |
| <b>:DL:</b>             | Delta line statistics                           | "            | <b>:Li:/:Ld:/:Lu:</b> | S      |
| <b>:Li:</b>             | Lines inserted by Delta                         | "            | <i>nnnnn</i>          | S      |
| <b>:Ld:</b>             | Lines deleted by Delta                          | "            | <i>nnnnn</i>          | S      |
| <b>:Lu:</b>             | Lines unchanged by Delta                        | "            | <i>nnnnn</i>          | S      |
| <b>:DT:</b>             | Delta type                                      | "            | <b>D or R</b>         | S      |
| <b>:I:</b>              | SCCS ID string (SID)                            | "            | See below**           | S      |
| <b>:R:</b>              | Release number                                  | "            | <i>nnnn</i>           | S      |
| <b>:L:</b>              | Level number                                    | "            | <i>nnnn</i>           | S      |
| <b>:B:</b>              | Branch number                                   | "            | <i>nnnn</i>           | S      |
| <b>:S:</b>              | Sequence number                                 | "            | <i>nnnn</i>           | S      |
| <b>:D:</b>              | Date delta created                              | "            | <b>:Dy:/:Dm:/:Dd:</b> | S      |
| <b>:Dy:</b>             | Year delta created                              | "            | <i>nn</i>             | S      |
| <b>:Dm:</b>             | Month delta created                             | "            | <i>nn</i>             | S      |
| <b>:Dd:</b>             | Day delta created                               | "            | <i>nn</i>             | S      |
| <b>:T:</b>              | Time delta created                              | "            | <b>:Th:::Tm:::Ts:</b> | S      |
| <b>:Th:</b>             | Hour delta created                              | "            | <i>nn</i>             | S      |
| <b>:Tm:</b>             | Minutes delta created                           | "            | <i>nn</i>             | S      |
| <b>:Ts:</b>             | Seconds delta created                           | "            | <i>nn</i>             | S      |
| <b>:P:</b>              | Programmer who created Delta                    | "            | <i>logname</i>        | S      |
| <b>:DS:</b>             | Delta sequence number                           | "            | <i>nnnn</i>           | S      |
| <b>:DP:</b>             | Predecessor Delta seq-no.                       | "            | <i>nnnn</i>           | S      |
| <b>:DI:</b>             | Seq-no. of deltas included, excluded or ignored | "            | <b>:Dn:/:Dx:/:Dg:</b> | S      |
| <b>:Dn:</b>             | Deltas included (seq #)                         | "            | <b>:DS: :DS:...</b>   | S      |
| <b>:Dx:</b>             | Deltas excluded (seq #)                         | "            | <b>:DS: :DS:...</b>   | S      |
| <b>:Dg:</b>             | Deltas ignored (seq #)                          | "            | <b>:DS: :DS:...</b>   | S      |
| <b>:MR:</b>             | MR numbers for delta                            | "            | <i>text</i>           | M      |
| <b>:C:</b>              | Comments for delta                              | "            | <i>text</i>           | M      |
| <b>:UN:</b>             | User names                                      | User Names   | <i>text</i>           | M      |
| <b>:FL:</b>             | Flag list                                       | Flags        | <i>text</i>           | M      |
| <b>:Y:</b>              | Module type flag                                | "            | <i>text</i>           | S      |
| <b>:MF:</b>             | MR validation flag                              | "            | <b>yes or no</b>      | S      |
| <b>:MP:</b>             | MR validation pgm name                          | "            | <i>text</i>           | S      |
| <b>:KF:</b>             | Keyword error, warning flag                     | "            | <b>yes or no</b>      | S      |
| <b>:KV:</b>             | Keyword validation string                       | "            | <i>text</i>           | S      |
| <b>:BF:</b>             | Branch flag                                     | "            | <b>yes or no</b>      | S      |

22768

22769

22770

22771

22772

22773

22774

22775

22776

22777

22778

22779

22780

22781

22782

22783

22784

22785

22786

| SCCS File Data Keywords |                              |              |                          |        |
|-------------------------|------------------------------|--------------|--------------------------|--------|
| Keyword                 | Data Item                    | File Section | Value                    | Format |
| <b>:J:</b>              | Joint edit flag              | "            | <b>yes or no</b>         | S      |
| <b>:LK:</b>             | Locked releases              | "            | <b>:R:...</b>            | S      |
| <b>:Q:</b>              | User-defined keyword         | "            | <i>text</i>              | S      |
| <b>:M:</b>              | Module name                  | "            | <i>text</i>              | S      |
| <b>:FB:</b>             | Floor boundary               | "            | <b>:R:</b>               | S      |
| <b>:CB:</b>             | Ceiling boundary             | "            | <b>:R:</b>               | S      |
| <b>:Ds:</b>             | Default SID                  | "            | <b>:I:</b>               | S      |
| <b>:ND:</b>             | Null delta flag              | "            | <b>yes or no</b>         | S      |
| <b>:FD:</b>             | File descriptive text        | Comments     | <i>text</i>              | M      |
| <b>:BD:</b>             | Body                         | Body         | <i>text</i>              | M      |
| <b>:GB:</b>             | Gotten body                  | "            | <i>text</i>              | M      |
| <b>:W:</b>              | A form of <i>what</i> string | N/A          | <b>:Z::M:\t:I:</b>       | S      |
| <b>:A:</b>              | A form of <i>what</i> string | N/A          | <b>:Z::Y: :M: :I::Z:</b> | S      |
| <b>:Z:</b>              | <i>what</i> string delimiter | N/A          | @(#)                     | S      |
| <b>:F:</b>              | SCCS filename                | N/A          | <i>text</i>              | S      |
| <b>:PN:</b>             | SCCS file pathname           | N/A          | <i>text</i>              | S      |

22787

\* :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:

22788

\*\* :R::L::B::S: if the delta is a branch delta (:BF: == yes)

22789

:R::L: if the delta is not a branch delta (:BF: == no)

**22790 STDERR**

22791 Used only for diagnostic messages.

**22792 OUTPUT FILES**

22793 None.

**22794 EXTENDED DESCRIPTION**

22795 None.

**22796 EXIT STATUS**

22797 The following exit values are returned:

22798 0 Successful completion.

22799 &gt;0 An error occurred.

**22800 CONSEQUENCES OF ERRORS**

22801 Default.

**22802 APPLICATION USAGE**

22803 None.

22804 **EXAMPLES**

22805 1. The following example:

22806 prs -d "User Names for :F: are:\n:UN:" s.file

22807 may write to standard output:

22808 User Names for s.file are:

22809 xyz

22810 131

22811 abc

22812 2. The following example:

22813 prs -d "Delta for pgm :M:: :I: - :D: By :P:" -r s.file

22814 may write to standard output:

22815 Delta for pgm main.c: 3.7 - 77/12/01 By cas

22816 3. As a special case:

22817 prs s.file

22818 may write to standard output:

22819 s.file:

22820 &lt;blank line&gt;

22821 D 1.1 77/12/01 00:00:00 cas 1 000000/00000/00000

22822 MRs:

22823 b178-12345

22824 b179-54321

22825 COMMENTS:

22826 this is the comment line for s.file initial delta

22827 &lt;blank line&gt;

22828 for each delta table entry of the *D* type. The only option allowed to be used with this  
22829 special case is the **-a** option.22830 **FUTURE DIRECTIONS**22831 A version of *prs* that fully supports the **XBD** specification, **Section 10.2, Utility Syntax**  
22832 **Guidelines** may be introduced in a future issue.22833 **SEE ALSO**22834 *admin, delta, get, what.*22835 **CHANGE HISTORY**

22836 First released in Issue 2.

22837 **Issue 4**

22838 Format reorganised.

22839 Exceptions to Utility Syntax Guidelines conformance noted.

22840 Internationalised environment variable support mandated.



22841 **Issue 5**

22842 The phrase “in which keyword substitution is followed by a newline” is deleted from the end of  
22843 the second paragraph of **Data Keywords** in the STDOUT section.

22844 The interpretation of the YY component of the **-c** *cutoff* argument is noted.

## 22845 NAME

22846 ps — report process status

## 22847 SYNOPSIS

22848 EX ps [-aA][*-defl*][*-G grouplist*][*-o format*]...[*-p proclist*][*-t termlist*]  
 22849 EX [*-U userlist*][*-g grouplist*][*-n namelist*][*-u userlist*]

## 22850 DESCRIPTION

22851 The *ps* utility writes information about processes, subject to having the appropriate privileges to  
 22852 obtain information about those processes.

22853 By default, *ps* selects all processes with the same effective user ID as the current user and the  
 22854 same controlling terminal as the invoker.

## 22855 OPTIONS

22856 The *ps* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

22857 The following options are supported:

22858 **-a** Write information for all processes associated with terminals. Implementations may  
 22859 omit session leaders from this list.

22860 **-A** Write information for all processes.

22861 EX **-d** Write information for all processes, except session leaders.

22862 EX **-e** Write information for all processes (equivalent to **-A**).

22863 EX **-f** Generate a *full* listing. (See the STDOUT section for the contents of a full listing.)

22864 EX **-g grouplist**  
 22865 Write information for processes whose session leaders are given in *grouplist*. The  
 22866 *grouplist* must be a single argument in the form of a blank- or comma-separated list.

22867 **-G grouplist**  
 22868 Write information for processes whose real group ID numbers are given in *grouplist*.  
 22869 The *grouplist* must be a single argument in the form of a blank- or comma-separated  
 22870 list.

22871 EX **-l** Generate a *long* listing. (See the STDOUT section for the contents of a long listing.)

22872 EX **-n namelist**  
 22873 Specify the name of an alternative system *namelist* file in place of the default. The name  
 22874 of the default file and the format of a *namelist* file are unspecified.

22875 **-o format**  
 22876 Write information according to the format specification given in *format*. This is fully  
 22877 described in the STDOUT section. Multiple **-o** options can be specified; the format  
 22878 specification will be interpreted as the space-character-separated concatenation of all  
 22879 the *format* option-arguments.

22880 **-p proclist**  
 22881 Write information for processes whose process ID numbers are given in *proclist*. The  
 22882 *proclist* must be a single argument in the form of a blank- or comma-separated list.

22883 **-t termlist**  
 22884 Write information for processes associated with terminals given in *termlist*. The *termlist*  
 22885 must be a single argument in the form of a blank- or comma-separated list. Terminal  
 22886 EX identifiers must be given in one of two forms: the device's filename (for example, **tty04**)  
 22887 or, if the device's filename starts with **tty**, just the identifier following the characters **tty**  
 22888 (for example, **04**).

22889 EX **-u *userlist***  
 22890 Write information for processes whose user ID numbers or login names are given in  
 22891 *userlist*. The *userlist* must be a single argument in the form of a blank- or comma-  
 22892 separated list. In the listing, the numerical user ID will be written unless the **-f** option  
 22893 is used, in which case the login name will be written.

22894 **-U *userlist***  
 22895 Write information for processes whose real user ID numbers or login names are given  
 22896 in *userlist*. The *userlist* must be a single argument in the form of a blank- or comma-  
 22897 separated list.

22898 With the exception of **-o *format***, all of the options shown are used to select processes. If any are  
 22899 specified, the default list will be ignored and *ps* will select the processes represented by the  
 22900 inclusive OR of all the selection-criteria options.

22901 **OPERANDS**  
 22902 None.

22903 **STDIN**  
 22904 Not used.

22905 **INPUT FILES**  
 22906 None.

22907 **ENVIRONMENT VARIABLES**  
 22908 The following environment variables affect the execution of *ps*:

22909 **COLUMNS**  
 22910 Override the system-selected horizontal screen size, used to determine the number of  
 22911 text columns to display. See the **XBD** specification, **Chapter 6, Environment Variables**  
 22912 for valid values and results when it is unset or null.

22913 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 22914 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 22915 default locale will be used. If any of the internationalisation variables contains an  
 22916 invalid setting, the utility will behave as if none of the variables had been defined.

22917 **LC\_ALL**  
 22918 If set to a non-empty string value, override the values of all the other  
 22919 internationalisation variables.

22920 **LC\_CTYPE**  
 22921 Determine the locale for the interpretation of sequences of bytes of text data as  
 22922 characters (for example, single- as opposed to multi-byte characters in arguments).

22923 **LC\_MESSAGES**  
 22924 Determine the locale that should be used to affect the format and contents of diagnostic  
 22925 messages written to standard error and informative messages written to standard  
 22926 output.

22927 EX **NLSPATH**  
 22928 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

22929 **LC\_TIME**  
 22930 Determine the format and contents of the date and time strings displayed.

22931 **ASYNCHRONOUS EVENTS**  
 22932 Default.

22933 **STDOUT**

22934 EX When the **-o** option is not specified, the standard output format is as follows. The column  
 22935 headings and descriptions of the columns in a *ps* listing are given below. The precise meanings  
 22936 of these fields are implementation-dependent. The letters **f** and **l** (below) indicate the option  
 22937 (**full** or **long**) that causes the corresponding heading to appear; **all** means that the heading  
 22938 always appears. Note that these two options determine only what information is provided for a  
 22939 process; they do not determine which processes will be listed.

|       |              |       |                                                                                                            |
|-------|--------------|-------|------------------------------------------------------------------------------------------------------------|
| 22940 | <b>F</b>     | (l)   | Flags (octal and additive) associated with the process.                                                    |
| 22941 | <b>S</b>     | (l)   | The state of the process.                                                                                  |
| 22942 | <b>UID</b>   | (f,l) | The user ID number of the process owner; the login name is printed<br>22943 under the <b>-f</b> option.    |
| 22944 | <b>PID</b>   | (all) | The process ID of the process; it is possible to kill a process if this<br>22945 datum is known.           |
| 22946 | <b>PPID</b>  | (f,l) | The process ID of the parent process.                                                                      |
| 22947 | <b>C</b>     | (f,l) | Processor utilisation for scheduling.                                                                      |
| 22948 | <b>PRI</b>   | (l)   | The priority of the process; higher numbers mean lower priority.                                           |
| 22949 | <b>NI</b>    | (l)   | Nice value; used in priority computation.                                                                  |
| 22950 | <b>ADDR</b>  | (l)   | The address of the process.                                                                                |
| 22951 | <b>SZ</b>    | (l)   | The size in blocks of the core image of the process.                                                       |
| 22952 | <b>WCHAN</b> | (l)   | The event for which the process is waiting or sleeping; if blank, the<br>22953 process is running.         |
| 22954 | <b>STIME</b> | (f)   | Starting time of the process.                                                                              |
| 22955 | <b>TTY</b>   | (all) | The controlling terminal for the process.                                                                  |
| 22956 | <b>TIME</b>  | (all) | The cumulative execution time for the process.                                                             |
| 22957 | <b>CMD</b>   | (all) | The command name; the full command name and its arguments are<br>22958 written under the <b>-f</b> option. |

22959 A process that has exited and has a parent, but has not yet been waited for by the parent, is  
 22960 marked **defunct**.

22961 Under the option **-f**, *ps* tries to determine the command name and arguments given when the  
 22962 process was created by examining memory or the swap area. Failing this, the command name,  
 22963 as it would appear without the option **-f**, is written in square brackets.

22964 The **-o** option allows the output format to be specified under user control.

22965 The format specification must be a list of names presented as a single argument, blank- or  
 22966 comma-separated. Each variable has a default header. The default header can be overridden by  
 22967 appending an equals sign and the new text of the header. The rest of the characters in the  
 22968 argument will be used as the header text. The fields specified will be written in the order  
 22969 specified on the command line, and should be arranged in columns in the output. The field  
 22970 widths will be selected by the system to be at least as wide as the header text (default or  
 22971 overridden value). If the header text is null, such as **-o user=**, the field width will be at least as  
 22972 wide as the default header text. If all header text fields are null, no header line will be written.

22973 The following names are recognised in the POSIX locale:

|       |              |                                                                                                                                                                            |
|-------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22974 | <b>ruser</b> | The real user ID of the process. This will be the textual user ID, if it can be obtained<br>22975 and the field width permits, or a decimal representation otherwise.      |
| 22976 | <b>user</b>  | The effective user ID of the process. This will be the textual user ID, if it can be<br>22977 obtained and the field width permits, or a decimal representation otherwise. |

|       |               |                                                                                                                |
|-------|---------------|----------------------------------------------------------------------------------------------------------------|
| 22978 | <b>rgroup</b> | The real group ID of the process. This will be the textual group ID, if it can be obtained                     |
| 22979 |               | and the field width permits, or a decimal representation otherwise.                                            |
| 22980 | <b>group</b>  | The effective group ID of the process. This will be the textual group ID, if it can be                         |
| 22981 |               | obtained and the field width permits, or a decimal representation otherwise.                                   |
| 22982 | <b>pid</b>    | The decimal value of the process ID.                                                                           |
| 22983 | <b>ppid</b>   | The decimal value of the parent process ID.                                                                    |
| 22984 | <b>pgid</b>   | The decimal value of the process group ID.                                                                     |
| 22985 | <b>pcpu</b>   | The ratio of CPU time used recently to CPU time available in the same period,                                  |
| 22986 |               | expressed as a percentage. The meaning of “recently” in this context is unspecified.                           |
| 22987 |               | The CPU time available is determined in an unspecified manner.                                                 |
| 22988 | <b>vsz</b>    | The size of the process in (virtual) memory in kilobytes as a decimal integer.                                 |
| 22989 | <b>nice</b>   | The decimal value of the system scheduling priority of the process. See <i>nice</i> .                          |
| 22990 | <b>etime</b>  | In the POSIX locale, the elapsed time since the process was started, in the form:                              |
| 22991 |               | [ [ <i>dd</i> -] <i>hh</i> : ] <i>mm</i> : <i>ss</i>                                                           |
| 22992 |               | where <i>dd</i> will represent the number of days, <i>hh</i> the number of hours, <i>mm</i> the number of      |
| 22993 |               | minutes, and <i>ss</i> the number of seconds. The <i>dd</i> field will be a decimal integer. The <i>hh</i> ,   |
| 22994 |               | <i>mm</i> and <i>ss</i> fields will be two-digit decimal integers padded on the left with zeros.               |
| 22995 | <b>time</b>   | In the POSIX locale, the cumulative CPU time of the process in the form:                                       |
| 22996 |               | [ [ <i>dd</i> -] <i>hh</i> : <i>mm</i> : <i>ss</i>                                                             |
| 22997 |               | The <i>dd</i> , <i>hh</i> , <i>mm</i> and <i>ss</i> fields will be as described in the <b>etime</b> specifier. |
| 22998 | <b>tty</b>    | The name of the controlling terminal of the process (if any) in the same format used by                        |
| 22999 |               | the <i>who</i> utility.                                                                                        |
| 23000 | <b>comm</b>   | The name of the command being executed ( <i>argv</i> [0] value) as a string.                                   |
| 23001 | <b>args</b>   | The command with all its arguments as a string. The implementation may truncate this                           |
| 23002 |               | value to the field width; it is implementation-dependent whether any further                                   |
| 23003 |               | truncation occurs. It is unspecified whether the string represented is a version of the                        |
| 23004 |               | argument list as it was passed to the command when it started, or is a version of the                          |
| 23005 |               | arguments as they may have been modified by the application. Applications cannot                               |
| 23006 |               | depend on being able to modify their argument list and having that modification be                             |
| 23007 |               | reflected in the output of <i>ps</i> .                                                                         |
| 23008 |               | Any field need not be meaningful in all implementations. In such a case a hyphen (–) should be                 |
| 23009 |               | output in place of the field value.                                                                            |
| 23010 |               | Only <b>comm</b> and <b>args</b> are allowed to contain blank characters; all others are not. Any              |
| 23011 |               | implementation-dependent variables will be specified in the system documentation along with                    |
| 23012 |               | the default header and indicating if the field may contain blank characters.                                   |
| 23013 |               | The following table specifies the default header to be used in the POSIX locale corresponding to               |
| 23014 |               | each format specifier.                                                                                         |

23015  
23016  
23017  
23018  
23019  
23020  
23021  
23022  
23023  
23024

| Format Specifier | Default Header | Format Specifier | Default Header |
|------------------|----------------|------------------|----------------|
| args             | COMMAND        | ppid             | PPID           |
| comm             | COMMAND        | rgroup           | RGROUP         |
| etime            | ELAPSED        | ruser            | RUSER          |
| group            | GROUP          | time             | TIME           |
| nice             | NI             | tty              | TT             |
| pcpu             | %CPU           | user             | USER           |
| pgid             | PGID           | vsz              | VSZ            |
| pid              | PID            |                  |                |

23025

**Table 3-14** Variable Names and Default Headers in *ps*

## 23026 **STDERR**

23027       Used only for diagnostic messages.

## 23028 **OUTPUT FILES**

23029       None.

## 23030 **EXTENDED DESCRIPTION**

23031       None.

## 23032 **EXIT STATUS**

23033       The following exit values are returned:

23034       0   Successful completion.

23035       >0   An error occurred.

## 23036 **CONSEQUENCES OF ERRORS**

23037       Default.

## 23038 **APPLICATION USAGE**

23039       Things can change while *ps* is running; the snapshot it gives is only true for an instant, and might not be accurate by the time it is displayed.

23041       The **args** format specifier is allowed to produce a truncated version of the command arguments. In some implementations, this information is no longer available when the *ps* utility is executed.

23043       If the field width is too narrow to display a textual ID, the system may use a numeric version. Normally, the system would be expected to choose large enough field widths, but if a large number of fields were selected to write, it might squeeze fields to their minimum sizes to fit on one line. One way to ensure adequate width for the textual IDs is to override the default header for a field to make it larger than most or all user or group names.

23048       There is no special quoting mechanism for header text. The header text is the rest of the argument. If multiple header changes are needed, multiple **-o** options can be used, such as:

23050       `ps -o "user=User Name" -o pid=Process\ ID`

23051       On some systems, especially multi-level secure systems, *ps* may be severely restricted and produce information only about child processes owned by the user.

23053 **EXAMPLES**

23054 The command:

23055 `ps -o user,pid,ppid=MOM -o args`

23056 writes at least the following in the POSIX locale: |

23057 

| USER | PID | MOM | COMMAND |
|------|-----|-----|---------|
|------|-----|-----|---------|

23058 

|        |    |    |                                             |
|--------|----|----|---------------------------------------------|
| helene | 34 | 12 | <code>ps -o uid,pid,ppid=MOM -o args</code> |
|--------|----|----|---------------------------------------------|

23059 The contents of the **COMMAND** field need not be the same in all implementations, due to  
23060 possible truncation.23061 **FUTURE DIRECTIONS**

23062 None.

23063 **SEE ALSO**23064 *kill, nice, renice.*23065 **CHANGE HISTORY**

23066 First released in Issue 2. |

23067 **Issue 4**

23068 Aligned with the ISO/IEC 9945-2: 1993 standard.

23069 **NAME**

23070           pwd — return working directory name

23071 **SYNOPSIS**

23072           pwd

23073 **DESCRIPTION**23074           The *pwd* utility will write an absolute pathname of the current working directory to standard  
23075           output.23076 **OPTIONS**

23077           None.

23078 **OPERANDS**

23079           None.

23080 **STDIN**

23081           Not used.

23082 **INPUT FILES**

23083           None.

23084 **ENVIRONMENT VARIABLES**23085           The following environment variables affect the execution of *pwd*:23086           *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
23087           *LANG* is unset or null, the corresponding value from the implementation-dependent  
23088           default locale will be used. If any of the internationalisation variables contains an  
23089           invalid setting, the utility will behave as if none of the variables had been defined.23090           *LC\_ALL*23091           If set to a non-empty string value, override the values of all the other  
23092           internationalisation variables.23093           *LC\_MESSAGES*23094           Determine the locale that should be used to affect the format and contents of diagnostic  
23095           messages written to standard error.23096 EX       *NLSPATH*23097           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.23098 **ASYNCHRONOUS EVENTS**

23099           Default.

23100 **STDOUT**23101           The *pwd* utility output will be an absolute pathname of the current working directory:

23102           "%s\n", &lt;directory pathname&gt;

23103 **STDERR**

23104           Used only for diagnostic messages.

23105 **OUTPUT FILES**

23106           None.

23107 **EXTENDED DESCRIPTION**

23108           None.



**23109 EXIT STATUS**

23110           The following exit values are returned:

23111           0   Successful completion.

23112           >0  An error occurred.

**23113 CONSEQUENCES OF ERRORS**

23114           If an error is detected, output will not be written to standard output, a diagnostic message will

23115           be written to standard error, and the exit status will not be zero.

**23116 APPLICATION USAGE**

23117           None.

**23118 EXAMPLES**

23119           None.

**23120 FUTURE DIRECTIONS**

23121           None.

**23122 SEE ALSO**

23123           *cd*, the **XSH** specification description of *getcwd()*.

**23124 CHANGE HISTORY**

23125           First released in Issue 2.

**23126 Issue 4**

23127           Aligned with the ISO/IEC 9945-2: 1993 standard.

23128 **NAME**

23129       read — read a line from standard input

23130 **SYNOPSIS**

23131       read [-r] var...

23132 **DESCRIPTION**23133       The *read* utility will read a single line from standard input.

23134       By default, unless the *-r* option is specified, backslash (\) acts as an escape character, as  
 23135       described in Section 2.2.1 on page 20. If standard input is a terminal device and the invoking  
 23136       shell is interactive, *read* will prompt for a continuation line when:

- 23137       • The shell reads an input line ending with a backslash, unless the *-r* option is specified.
- 23138       • A here-document is not terminated after a newline character is entered.

23139       The line will be split into fields as in the shell (see Section 2.6.5 on page 38); the first field will be  
 23140       assigned to the first variable *var*, the second field to the second variable *var*, and so on. If there  
 23141       are fewer *var* operands specified than there are fields, the leftover fields and their intervening  
 23142       separators will be assigned to the last *var*. If there are fewer fields than *vars*, the remaining *vars*  
 23143       will be set to empty strings.

23144       The setting of variables specified by the *var* operands will affect the current shell execution  
 23145       environment; see Section 2.12 on page 63. If it is called in a subshell or separate utility execution  
 23146       environment, such as one of the following:

```
23147 (read foo)
23148 nohup read ...
23149 find . -exec read ... \;
```

23150       it will not affect the shell variables in the caller's environment.

23151 **OPTIONS**23152       The *read* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

23153       The following option is supported:

23154       *-r*       Do not treat a backslash character in any special way. Consider each backslash to be  
 23155       part of the input line.

23156 **OPERANDS**

23157       The following operands are supported:

23158       *var*       The name of an existing or non-existing shell variable.

23159 **STDIN**

23160       The standard input must be a text file.

23161 **INPUT FILES**

23162       None.

23163 **ENVIRONMENT VARIABLES**23164       The following environment variables affect the execution of *read*:

23165       *IFS*       Determine the internal field separators used to delimit fields. See Section 2.5.3 on page  
 23166       29.

23167       *LANG*      Provide a default value for the internationalisation variables that are unset or null. If  
 23168       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 23169       default locale will be used. If any of the internationalisation variables contains an  
 23170       invalid setting, the utility will behave as if none of the variables had been defined.

23171 **LC\_ALL**  
 23172 If set to a non-empty string value, override the values of all the other  
 23173 internationalisation variables.

23174 **LC\_CTYPE**  
 23175 Determine the locale for the interpretation of sequences of bytes of text data as  
 23176 characters (for example, single- as opposed to multi-byte characters in arguments).

23177 **LC\_MESSAGES**  
 23178 Determine the locale that should be used to affect the format and contents of diagnostic  
 23179 messages written to standard error.

23180 EX **NLSPATH**  
 23181 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

23182 **PS2** Provide the prompt string that an interactive shell will write to standard error when a  
 23183 line ending with a backslash is read and the **-r** option was not specified, or if a here-  
 23184 document is not terminated after a newline character is entered.

23185 **ASYNCHRONOUS EVENTS**  
 23186 Default.

23187 **STDOUT**  
 23188 Not used.

23189 **STDERR**  
 23190 Used for diagnostic messages and prompts for continued input.

23191 **OUTPUT FILES**  
 23192 None.

23193 **EXTENDED DESCRIPTION**  
 23194 None.

23195 **EXIT STATUS**  
 23196 The following exit values are returned:  
 23197 0 Successful completion.  
 23198 >0 End-of-file was detected or an error occurred.

23199 **CONSEQUENCES OF ERRORS**  
 23200 Default.

23201 **APPLICATION USAGE**  
 23202 The *read* utility has historically been a shell built-in.

23203 The **-r** option is included to enable *read* to subsume the purpose of the *line* utility, which has  
 23204 been marked LEGACY.

23205 The results are undefined if an end-of-file is detected following a backslash at the end of a line  
 23206 when **-r** is not specified.

23207 **EXAMPLES**  
 23208 The following command:  
 23209 

```
while read -r xx yy
```

  
 23210 

```
do
```

  
 23211 

```
 printf "%s %s\n" "$yy" "$xx"
```

  
 23212 

```
done < input_file
```

  
 23213 prints a file with the first field of each line moved to the end of the line.

23214 **FUTURE DIRECTIONS**

23215 None.

23216 **SEE ALSO**

23217 None.

23218 **CHANGE HISTORY**

23219 First released in Issue 2.

23220 **Issue 4**23221 Relocated from the *sh* description for alignment with the ISO/IEC 9945-2: 1993 standard.

## 23222 NAME

23223 renice — set system scheduling priorities of running processes

## 23224 SYNOPSIS

23225 renice [-n *increment*][ -g | -p | -u ] *ID*...

23226 OB renice *nice\_value*[-p] *pid*...[-g *gid*...][-p *pid*...][-u *user*...]

23227 OB renice *nice\_value* -g *gid*...[-g *gid*...][-p *pid*...][-u *user*...]

23228 OB renice *nice\_value* -u *user*...[-g *gid*...][-p *pid*...][-u *user*...]

## 23229 DESCRIPTION

23230 The *renice* utility requests that the system scheduling priorities (see the definition of **system**  
 23231 **scheduling priority** in the **XBD** specification, **Chapter 2, Glossary**) of one or more running  
 23232 processes be changed. By default, the applicable processes are specified by their process IDs.  
 23233 When a process group is specified (see -g), the request applies to all processes in the process  
 23234 group.

23235 The system scheduling priority is bounded in an implementation-dependent manner. If the  
 23236 OB requested *increment* (or *nice\_value* in the obsolescent versions) would raise or lower the system  
 23237 scheduling priority of the executed utility beyond implementation-dependent limits, then the  
 23238 limit whose value was exceeded is used.

23239 FIPS When a user is *reniced*, the request applies to all processes whose saved set-user-ID matches the  
 23240 user ID corresponding to the user.

23241 Regardless of which options are supplied or any other factor, *renice* will not alter the system  
 23242 scheduling priorities of any process unless the user requesting such a change has appropriate  
 23243 privileges to do so for the specified process. If the user lacks appropriate privileges to perform  
 23244 the requested action, the utility will return an error status.

23245 FIPS The saved set-user-ID of the user's process will be checked instead of its effective user ID when  
 23246 *renice* attempts to determine the user ID of the process in order to determine whether the user  
 23247 has appropriate privileges.

## 23248 OPTIONS

23249 OB The *renice* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 23250 obsolescent version conforms with the following exceptions:

23251 • The first operand, *nice\_value*, must precede the options and can have the appearance of a  
 23252 multi-digit option.

23253 • The -g, -p and -u options can each take multiple option-arguments.

23254 • The *pid* option-argument can be used without its -p option.

23255 The following options are supported:

23256 OB -g Interpret all operands (or just the *gid* arguments in the obsolescent version) as unsigned  
 23257 decimal integer process group IDs.

23258 -n *increment*

23259 Specify how the system scheduling priority of the specified process or processes is to  
 23260 be adjusted. The *increment* option-argument is a positive or negative decimal integer  
 23261 that will be used to modify the system scheduling priority of the specified process or  
 23262 processes.

23263 Positive *increment* values cause a lower system scheduling priority. Negative *increment*  
 23264 values may require appropriate privileges and will cause a higher system scheduling  
 23265 priority.

23266 OB      **-p**      Interpret all operands (or just the *pid* arguments in the obsolescent version) as unsigned decimal integer process IDs. The **-p** option is the default if no options are specified.  
23267

23268 OB      **-u**      Interpret all operands (or just the *user* arguments in the obsolescent version) as users. If a user exists with a user name equal to the operand, then the user ID of that user will be used in further processing. Otherwise, if the operand represents an unsigned decimal integer, it will be used as the numeric user ID of the user.  
23269  
23270  
23271

## 23272 OPERANDS

23273      The following operands are supported:

23274      **ID**      A process ID, process group ID or user name/user ID, depending on the option selected.  
23275

23276 OB      **nice\_value**  
23277      The value specified is taken as the actual system scheduling priority, rather than as an increment to the existing system scheduling priority. Specifying a scheduling priority higher than that of the existing process may require appropriate privileges.  
23278  
23279

## 23280 STDIN

23281      Not used.

## 23282 INPUT FILES

23283      None.

## 23284 ENVIRONMENT VARIABLES

23285      The following environment variables affect the execution of *renice*:

23286      **LANG**      Provide a default value for the internationalisation variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.  
23287  
23288  
23289

23290      **LC\_ALL**  
23291      If set to a non-empty string value, override the values of all the other internationalisation variables.  
23292

23293      **LC\_CTYPE**  
23294      Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments).  
23295

23296      **LC\_MESSAGES**  
23297      Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.  
23298

23299 EX      **NLSPATH**  
23300      Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 23301 ASYNCHRONOUS EVENTS

23302      Default.

## 23303 STDOUT

23304      Not used.

## 23305 STDERR

23306      Used only for diagnostic messages.

## 23307 OUTPUT FILES

23308      None.

23309 **EXTENDED DESCRIPTION**

23310 None.

23311 **EXIT STATUS**

23312 The following exit values are returned:

23313 0 Successful completion.

23314 &gt;0 An error occurred.

23315 **CONSEQUENCES OF ERRORS**

23316 Default.

23317 **APPLICATION USAGE**

23318 None.

23319 **EXAMPLES**

23320 1. Adjust the system scheduling priority so that process IDs 987 and 32 would have a lower  
 23321 scheduling priority:

23322 `renice -n 5 -p 987 32`

23323 2. Adjust the system scheduling priority so that group IDs 324 and 76 would have a higher  
 23324 scheduling priority, if the user has the appropriate privileges to do so:

23325 `renice -n -4 -g 324 76`

23326 3. Adjust the system scheduling priority so that numeric user ID 8 and user **sas** would have a  
 23327 lower scheduling priority:

23328 `renice -n 4 -u 8 sas`

23329 Useful nice values on historical systems include 19 or 20 (the affected processes will run only  
 23330 when nothing else in the system attempts to run), 0 (the base scheduling priority), and any  
 23331 negative number (to make processes run faster).

23332 **FUTURE DIRECTIONS**

23333 None.

23334 **SEE ALSO**23335 *nice*.23336 **CHANGE HISTORY**

23337 First released in Issue 4.

23338 **Issue 5**23339 In the SYNOPSIS, an ellipsis is added to the **-u** option in all three obsolescent forms.

## 23340 NAME

23341       rm — remove directory entries

## 23342 SYNOPSIS

23343       rm [-fiRr] *file*...

## 23344 DESCRIPTION

23345       The *rm* utility removes the directory entry specified by each *file* argument.

23346       If either of the files dot or dot-dot are specified as the basename portion of an operand (that is,  
23347       the final pathname component), *rm* will write a diagnostic message to standard error and do  
23348       nothing more with such operands.

23349       For each *file* the following steps are taken:

- 23350       1. If the *file* does not exist:
  - 23351           a. If the **-f** option is not specified, write a diagnostic message to standard error.
  - 23352           b. Go on to any remaining *files*.
- 23353       2. If *file* is of type directory, the following steps are taken:
  - 23354           a. If neither the **-R** option nor the **-r** option is specified, write a diagnostic message to  
23355           standard error, do nothing more with *file*, and go on to any remaining files.
  - 23356           b. If the **-f** option is not specified, and either the permissions of *file* do not permit  
23357           writing and the standard input is a terminal or the **-i** option is specified, write a  
23358           prompt to standard error and read a line from the standard input. If the response is  
23359           not affirmative, do nothing more with the current file and go on to any remaining  
23360           files.
  - 23361           c. For each entry contained in *file*, other than dot or dot-dot, the four steps listed here  
23362           (1-4) are taken with the entry as if it were a *file* operand.
  - 23363           d. If the **-i** option is specified, write a prompt to standard error and read a line from the  
23364           standard input. If the response is not affirmative, do nothing more with the current  
23365           file, and go on to any remaining files.
- 23366       3. If *file* is not of type directory, the **-f** option is not specified, and either the permissions of  
23367       *file* do not permit writing and the standard input is a terminal or the **-i** option is specified,  
23368       write a prompt to the standard error and read a line from the standard input. If the  
23369       response is not affirmative, do nothing more with the current file and go on to any  
23370       remaining files.
- 23371       4. If the current file is a directory, *rm* will perform actions equivalent to the **XSH** specification  
23372       *rmdir()* function called with a pathname of the current file used as the *path* argument. If  
23373       the current file is not a directory, *rm* will perform actions equivalent to the **XSH**  
23374       specification *unlink()* function called with a pathname of the current file used as the *path*  
23375       argument.

23376       If this fails for any reason, *rm* will write a diagnostic message to standard error, do nothing  
23377       more with the current file, and go on to any remaining files.

23378       The *rm* utility is able to descend to arbitrary depths in a file hierarchy, and will not fail due to  
23379       path length limitations (unless an operand specified by the user exceeds system limitations).

## 23380 OPTIONS

23381       The *rm* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
23382       following options are supported:



|       |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
|-------|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 23383 | <b>-f</b>                    | Do not prompt for confirmation. Do not write diagnostic messages or modify the exit status in the case of non-existent operands. Any previous occurrences of the <b>-i</b> option will be ignored.                                                                                                                                                             |  |
| 23384 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23385 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23386 | <b>-i</b>                    | Prompt for confirmation as described previously. Any previous occurrences of the <b>-f</b> option will be ignored.                                                                                                                                                                                                                                             |  |
| 23387 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23388 | <b>-R</b>                    | Remove file hierarchies. See the DESCRIPTION section.                                                                                                                                                                                                                                                                                                          |  |
| 23389 | <b>-r</b>                    | Equivalent to <b>-R</b> .                                                                                                                                                                                                                                                                                                                                      |  |
| 23390 | <b>OPERANDS</b>              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23391 |                              | The following operand is supported:                                                                                                                                                                                                                                                                                                                            |  |
| 23392 | <i>file</i>                  | A pathname of a directory entry to be removed.                                                                                                                                                                                                                                                                                                                 |  |
| 23393 | <b>STDIN</b>                 |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23394 |                              | Used to read an input line in response to each prompt specified in the STDOUT section.                                                                                                                                                                                                                                                                         |  |
| 23395 |                              | Otherwise, the standard input will not be used.                                                                                                                                                                                                                                                                                                                |  |
| 23396 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23397 |                              | None.                                                                                                                                                                                                                                                                                                                                                          |  |
| 23398 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23399 |                              | The following environment variables affect the execution of <i>rm</i> :                                                                                                                                                                                                                                                                                        |  |
| 23400 | <b>LANG</b>                  | Provide a default value for the internationalisation variables that are unset or null. If <i>LANG</i> is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined. |  |
| 23401 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23402 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23403 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23404 | <b>LC_ALL</b>                |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23405 |                              | If set to a non-empty string value, override the values of all the other internationalisation variables.                                                                                                                                                                                                                                                       |  |
| 23406 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23407 | <b>LC_COLLATE</b>            |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23408 |                              | Determine the locale for the behaviour of ranges, equivalence classes, and multi-character collating elements used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the LC_MESSAGES category.                                                                                                                               |  |
| 23409 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23410 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23411 | <b>LC_CTYPE</b>              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23412 |                              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- versus multi-byte characters in arguments) and the behaviour of character classes within regular expressions used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the LC_MESSAGES category.             |  |
| 23413 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23414 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23415 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23416 | <b>LC_MESSAGES</b>           |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23417 |                              | Determine the locale for the processing of affirmative responses that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                       |  |
| 23418 |                              |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23419 | EX                           | <b>NLSPATH</b>                                                                                                                                                                                                                                                                                                                                                 |  |
| 23420 |                              | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                        |  |
| 23421 | <b>ASYNCHRONOUS EVENTS</b>   |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23422 |                              | Default.                                                                                                                                                                                                                                                                                                                                                       |  |
| 23423 | <b>STDOUT</b>                |                                                                                                                                                                                                                                                                                                                                                                |  |
| 23424 |                              | Not used.                                                                                                                                                                                                                                                                                                                                                      |  |

23425 **STDERR**

23426 Prompts are written to standard error under the conditions specified in the DESCRIPTION and  
23427 OPTIONS sections. The prompts will contain the *file* pathname, but their format is otherwise  
23428 unspecified. The standard error is also used for diagnostic messages.

23429 **OUTPUT FILES**

23430 None.

23431 **EXTENDED DESCRIPTION**

23432 None.

23433 **EXIT STATUS**

23434 The following exit values are returned:

23435 0 If the **-f** option was not specified, all the named directory entries were removed; otherwise,  
23436 all the existing named directory entries were removed.  
23437 >0 An error occurred.

23438 **CONSEQUENCES OF ERRORS**

23439 Default.

23440 **APPLICATION USAGE**

23441 The *rm* utility is forbidden to remove the names *dot* and *dot-dot* in order to avoid the  
23442 consequences of inadvertently doing something like:

23443 `rm -r .*`

23444 Some systems do not permit the removal of the last link to an executable binary file that is being  
23445 executed; see the [EBUSY] error in the **XSH** specification *unlink()* description. Thus, the *rm*  
23446 utility can fail to remove such files.

23447 The **-i** option causes *rm* to prompt and read the standard input even if the standard input is not  
23448 a terminal, but in the absence of **-i** the mode prompting is not done when the standard input is  
23449 not a terminal.

23450 **EXAMPLES**

23451 1. The following command:

23452 `rm a.out core`

23453 removes the directory entries: **a.out** and **core**.

23454 2. The following command:

23455 `rm -Rf junk`

23456 removes the directory **junk** and all its contents, without prompting.

23457 **FUTURE DIRECTIONS**

23458 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
23459 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
23460 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
23461 finalised.

23462 **SEE ALSO**

23463 *rmdir*, the **XSH** specification description of *remove()*, *unlink()*.

23464 **CHANGE HISTORY**

23465 First released in Issue 2.

23466 **Issue 4**

23467 Aligned with the ISO/IEC 9945-2: 1993 standard.

23468 **Issue 5**

23469 FUTURE DIRECTIONS section added.

## 23470 NAME

23471 `rmdel` — remove a delta from an SCCS file (**DEVELOPMENT**)

## 23472 SYNOPSIS

23473 EX `rmdel -r SID file...`

## 23474 DESCRIPTION

23475 The `rmdel` utility removes the delta specified by the SID from each named SCCS file. The delta to  
 23476 be removed must be the most recent delta in its branch in the delta chain of each named SCCS  
 23477 file. In addition, the SID specified must not be that of a version being edited for the purpose of  
 23478 making a delta; that is, if a *p-file* (see *get*) exists for the named SCCS file, the SID specified must  
 23479 not appear in any entry of the *p-file*.

23480 Removal of a delta is restricted to:

- 23481 1. the user who made the delta
- 23482 2. the owner of the SCCS file
- 23483 3. the owner of the directory containing the SCCS file.

## 23484 OPTIONS

23485 The `rmdel` utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 23486 following option is supported:

23487 `-r SID` Specify the SCCS identification string (*SID*) of the delta to be deleted.

## 23488 OPERANDS

23489 The following operands are supported:

23490 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, `rmdel` behaves  
 23491 as though each file in the directory were specified as a named file, except that non-  
 23492 SCCS files (last component of the pathname does not begin with *s*.) and unreadable  
 23493 files are silently ignored.

23494 If a single instance *file* is specified as `-`, the standard input is read; each line of the  
 23495 standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files  
 23496 and unreadable files are silently ignored.

## 23497 STDIN

23498 The standard input is a text file used only when the *file* operand is specified as `-`. Each line of the  
 23499 text file is interpreted as an SCCS pathname.

## 23500 INPUT FILES

23501 The SCCS files are files of unspecified format.

## 23502 ENVIRONMENT VARIABLES

23503 The following environment variables affect the execution of `rmdel`:

23504 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 23505 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 23506 default locale will be used. If any of the internationalisation variables contains an  
 23507 invalid setting, the utility will behave as if none of the variables had been defined.

23508 *LC\_ALL*

23509 If set to a non-empty string value, override the values of all the other  
 23510 internationalisation variables.

23511 *LC\_CTYPE*

23512 Determine the locale for the interpretation of sequences of bytes of text data as  
 23513 characters (for example, single- as opposed to multi-byte characters in arguments and

23514 input files).

23515 **LC\_MESSAGES**

23516 Determine the locale that should be used to affect the format and contents of diagnostic

23517 messages written to standard error.

23518 **NLSPATH**

23519 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

23520 **ASYNCHRONOUS EVENTS**

23521 Default.

23522 **STDOUT**

23523 Not used.

23524 **STDERR**

23525 Used only for diagnostic messages.

23526 **OUTPUT FILES**

23527 The SCCS files are files of unspecified format. During processing of a *file*, a temporary *x-file*, as

23528 described in *admin*, may be created and deleted; a locking *z-file*, as described in *get*, may be

23529 created and deleted.

23530 **EXTENDED DESCRIPTION**

23531 None.

23532 **EXIT STATUS**

23533 The following exit values are returned:

23534 0 Successful completion.

23535 >0 An error occurred.

23536 **CONSEQUENCES OF ERRORS**

23537 Default.

23538 **APPLICATION USAGE**

23539 None.

23540 **EXAMPLES**

23541 None.

23542 **FUTURE DIRECTIONS**

23543 None.

23544 **SEE ALSO**

23545 *delta*, *get*, *prs*.

23546 **CHANGE HISTORY**

23547 First released in Issue 2.

23548 **Issue 4**

23549 Format reorganised.

23550 Utility Syntax Guidelines support mandated.

23551 Internationalised environment variable support mandated.

23552 **NAME**

23553           rmdir — remove directories

23554 **SYNOPSIS**23555           rmdir [-p] *dir*...23556 **DESCRIPTION**23557           The *rmdir* utility will remove the directory entry specified by each *dir* operand, which must refer  
23558           to an empty directory.23559           Directories will be processed in the order specified. If a directory and a subdirectory of that  
23560           directory are specified in a single invocation of the *rmdir* utility, the subdirectory must be  
23561           specified before the parent directory so that the parent directory will be empty when the *rmdir*  
23562           utility tries to remove it.23563 **OPTIONS**23564           The *rmdir* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

23565           The following option is supported:

23566           **-p**       Remove all directories in a pathname. For each *dir* operand:

- 23567                       1. The directory entry it names will be removed.
- 
- 23568                       2. If the
- dir*
- operand includes more than one pathname component, effects
- 
- 23569                       equivalent to the following command will occur:

23570                       rmdir -p \$(dirname *dir*)23571 **OPERANDS**

23572           The following operand is supported:

23573           *dir*       A pathname of an empty directory to be removed.23574 **STDIN**

23575           Not used.

23576 **INPUT FILES**

23577           None.

23578 **ENVIRONMENT VARIABLES**23579           The following environment variables affect the execution of *rmdir*:23580           **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
23581           **LANG** is unset or null, the corresponding value from the implementation-dependent  
23582           default locale will be used. If any of the internationalisation variables contains an  
23583           invalid setting, the utility will behave as if none of the variables had been defined.23584           **LC\_ALL**23585                       If set to a non-empty string value, override the values of all the other  
23586                       internationalisation variables.23587           **LC\_CTYPE**23588                       Determine the locale for the interpretation of sequences of bytes of text data as  
23589                       characters (for example, single- as opposed to multi-byte characters in arguments).23590           **LC\_MESSAGES**23591                       Determine the locale that should be used to affect the format and contents of diagnostic  
23592                       messages written to standard error.23593 **EX**       **NLSPATH**23594                       Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

23595 **ASYNCHRONOUS EVENTS**

23596 Default.

23597 **STDOUT**

23598 Not used.

23599 **STDERR**

23600 Used only for diagnostic messages.

23601 **OUTPUT FILES**

23602 None.

23603 **EXTENDED DESCRIPTION**

23604 None.

23605 **EXIT STATUS**

23606 The following exit values are returned:

23607 0 Each directory entry specified by a *dir* operand was removed successfully.

23608 &gt;0 An error occurred.

23609 **CONSEQUENCES OF ERRORS**

23610 Default.

23611 **APPLICATION USAGE**23612 The definition of an empty directory is one that contains, at most, directory entries for dot and  
23613 dot-dot.23614 **EXAMPLES**23615 If a directory **a** in the current directory is empty except it contains a directory **b** and **a/b** is empty  
23616 except it contains a directory **c**:23617 `rmdir -p a/b/c`

23618 will remove all three directories.

23619 **FUTURE DIRECTIONS**

23620 None.

23621 **SEE ALSO**23622 *rm*, the XSH specification description of *remove()*, *rmdir()*, *unlink()*.23623 **CHANGE HISTORY**

23624 First released in Issue 2.

23625 **Issue 4**23626 Separated from the *rm* description and aligned with the ISO/IEC 9945-2: 1993 standard.

23627 **NAME**23628           sact — print current SCCS file-editing activity (**DEVELOPMENT**)23629 **SYNOPSIS**23630 EX       sact *file...*23631 **DESCRIPTION**

23632       The *sact* utility informs the user of any impending deltas to a named SCCS file by writing a list to  
 23633       standard output. This situation occurs when *get -e* has been executed previously without a  
 23634       subsequent execution of *delta*.

23635 **OPTIONS**

23636       None.

23637 **OPERANDS**

23638       The following operand is supported:

23639       *file*     A pathname of an existing SCCS file or a directory. If *file* is a directory, *sact* behaves as  
 23640       though each file in the directory were specified as a named file, except that non-SCCS  
 23641       files (last component of the pathname does not begin with s.) and unreadable files are  
 23642       silently ignored.

23643       If a single instance *file* is specified as *-*, the standard input is read; each line of the  
 23644       standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files  
 23645       and unreadable files are silently ignored.

23646 **STDIN**

23647       The standard input is a text file used only when the *file* operand is specified as *-*. Each line of the  
 23648       text file is interpreted as an SCCS pathname.

23649 **INPUT FILES**

23650       Any SCCS files interrogated are files of an unspecified format.

23651 **ENVIRONMENT VARIABLES**23652       The following environment variables affect the execution of *sact*:

23653       *LANG*     Provide a default value for the internationalisation variables that are unset or null. If  
 23654       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 23655       default locale will be used. If any of the internationalisation variables contains an  
 23656       invalid setting, the utility will behave as if none of the variables had been defined.

23657       *LC\_ALL*

23658       If set to a non-empty string value, override the values of all the other  
 23659       internationalisation variables.

23660       *LC\_CTYPE*

23661       Determine the locale for the interpretation of sequences of bytes of text data as  
 23662       characters (for example, single- as opposed to multi-byte characters in arguments and  
 23663       input files).

23664       *LC\_MESSAGES*

23665       Determine the locale that should be used to affect the format and contents of diagnostic  
 23666       messages written to standard error.

23667       *NLSPATH*23668       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.23669 **ASYNCHRONOUS EVENTS**

23670       Default.



**23671 STDOUT**

23672 The output for each named file consists of a line in the following format:

23673 "%sΔ%sΔ%sΔ%sΔ%s\n", <SID>, <new SID>, <login>, <date>, <time>

23674 <SID> Specifies the SID of a delta that currently exists in the SCCS file to which changes will  
23675 be made to make the new delta.

23676 <new SID>

23677 Specifies the SID for the new delta to be created.

23678 <login> Contains the login name of the user who will make the delta (that is, who executed a  
23679 *get* for editing).

23680 <date> Contains the date that *get -e* was executed, in the format used by the *prs :D:* data  
23681 keyword.

23682 <time> Contains the time that *get -e* was executed, in the format used by the *prs :T:* data  
23683 keyword.

23684 If there is more than one named file or if a directory or standard input is named, each pathname  
23685 is written before each of the preceding lines:

23686 "\n%s:\n", <pathname>

**23687 STDERR**

23688 Used only for optional informative messages concerning SCCS files with no impending deltas,  
23689 and for diagnostic messages.

**23690 OUTPUT FILES**

23691 None.

**23692 EXTENDED DESCRIPTION**

23693 None.

**23694 EXIT STATUS**

23695 The following exit values are returned:

23696 0 Successful completion.

23697 >0 An error occurred.

**23698 CONSEQUENCES OF ERRORS**

23699 Default.

**23700 APPLICATION USAGE**

23701 None.

**23702 EXAMPLES**

23703 None.

**23704 FUTURE DIRECTIONS**

23705 None.

**23706 SEE ALSO**

23707 *delta*, *get*, *unget*.

23708 **CHANGE HISTORY**

23709 First released in Issue 2. |

23710 **Issue 4**

23711 Format reorganised.

23712 Utility Syntax Guidelines support mandated.

23713 Internationalised environment variable support mandated.

23714 **Issue 4, Version 2**23715 The STDERR section encompasses informative messages concerning SCCS files with no |  
23716 impending deltas.

23717 **NAME**23718 `sccs` — front end for the SCCS subsystem (**DEVELOPMENT**)23719 **SYNOPSIS**23720 EX `sccs [-r][-d path][-p path] command [options...][operands...]`23721 **DESCRIPTION**23722 The `sccs` utility is a front end to the SCCS programs. It also includes the capability to run set-user-id to another user to provide additional protection.23724 The `sccs` utility invokes the specified *command* with the specified *options* and *operands*. By default, each of the *operands* is modified by prefixing it with the string `SCCS/s..`.23726 The *command* operand can be one of the SCCS utilities in this specification (*admin*, *delta*, *get*, *prs*, *rm del*, *sact*, *unget*, *val* or *what*) or one of the pseudo-utilities listed in the EXTENDED DESCRIPTION section.23729 **OPTIONS**23730 The `sccs` utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except that *options* operands are actually options to be passed to the utility named by *command*. When the portion of the command:23733 `command [options ...] [operands ...]`23734 is considered, all of the pseudo-utilities used as *command* support the Utility Syntax Guidelines. Any of the other SCCS utilities that can be invoked in this manner support the Guidelines to the extent indicated by their individual OPTIONS sections.23737 The following options are supported preceding the *command* operand:23738 **-d path** A pathname of a directory to be used as a root directory for the SCCS files. The default is the current directory. The **-d** option takes precedence over the *PROJECTDIR* variable. See **-p**.23741 **-p path** A pathname of a directory in which the SCCS files are located. The default is the **SCCS** directory.23743 The **-p** option differs from the **-d** option in that the **-d** option-argument is prefixed to the entire pathname and the **-p** option-argument is inserted before the final component of the pathname. For example:23746 `sccs -d /x -p y get a/b`

23747 will convert to:

23748 `get /x/a/y/s.b`

23749 This allows the creation of aliases such as:

23750 `alias syssccs="sccs -d /usr/src"`

23751 which will be used as:

23752 `syssccs get cmd/who.c`23753 **-r** Invoke *command* with the real user ID of the process, not any effective user ID that the `sccs` utility is set to. Certain commands (*admin*, *check*, *clean*, *diffs*, *info*, *rm del* and *tell*) cannot be run set-user-ID by all users, since this would allow anyone to change the authorisations. These commands are always run as the real user.

23757 **OPERANDS**

23758 The following operands are supported:

23759 *command*23760 An SCCS utility name or the name of one of the pseudo-utilities listed in the  
23761 EXTENDED DESCRIPTION section.23762 *options* An option or option-argument to be passed to *command*.23763 *operands*23764 An operand to be passed to *command*.23765 **STDIN**23766 See the utility description for the specified *command*.23767 **INPUT FILES**23768 See the utility description for the specified *command*.23769 **ENVIRONMENT VARIABLES**23770 The following environment variables affect the execution of *sccs*:23771 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
23772 *LANG* is unset or null, the corresponding value from the implementation-dependent  
23773 default locale will be used. If any of the internationalisation variables contains an  
23774 invalid setting, the utility will behave as if none of the variables had been defined.23775 *LC\_ALL*23776 If set to a non-empty string value, override the values of all the other  
23777 internationalisation variables.23778 *LC\_CTYPE*23779 Determine the locale for the interpretation of sequences of bytes of text data as  
23780 characters (for example, single- as opposed to multi-byte characters in arguments and  
23781 input files).23782 *LC\_MESSAGES*23783 Determine the locale that should be used to affect the format and contents of diagnostic  
23784 messages written to standard error.23785 *NLSPATH*23786 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.23787 *PROJECTDIR*23788 Provide a default value for the *-d path* option. If the value of *PROJECTDIR* begins  
23789 with a slash, it is considered an absolute pathname; otherwise, the home directory of a  
23790 user of that name is examined for a subdirectory *src* or *source*. If such a directory is  
23791 found, it is used. Otherwise, the value is used as a relative pathname.23792 Additional environment variable effects may be found in the utility description for the specified  
23793 *command*.23794 **ASYNCHRONOUS EVENTS**

23795 Default.

23796 **STDOUT**23797 See the utility description for the specified *command*.23798 **STDERR**23799 See the utility description for the specified *command*.

23800 **OUTPUT FILES**

23801        See the utility description for the specified *command*.

23802 **EXTENDED DESCRIPTION**

23803        The following pseudo-utilities are supported as *command* operands. All options referred to in  
23804        the following list are values given in the *options* operands following *command*.

23805        **check**    Equivalent to **info**, except that nothing is printed if nothing is being edited, and a non-  
23806        zero exit status is returned if anything is being edited. The intent is to have this  
23807        included in an “install” entry in a makefile to ensure that everything is included into  
23808        the SCCS file before a version is installed.

23809        **clean**    Remove everything from the current directory that can be recreated from SCCS files,  
23810        but do not remove any files being edited. If the **-b** option is given, branches are  
23811        ignored in the determination of whether they are being edited; this is dangerous if  
23812        branches are kept in the same directory.

23813        **create**    Create an SCCS file, taking the initial contents from the file of the same name. Any  
23814        options to *admin* are accepted. If the creation is successful, the original files are  
23815        renamed by prefixing the basenames with a comma. These renamed files should be  
23816        removed after it has been verified that the SCCS files have been created successfully.

23817        **delget**    Perform a *delta* on the named files and then *get* new versions. The new versions will  
23818        have ID keywords expanded and will not be editable. Any **-m**, **-p**, **-r**, **-s**, and **-y**  
23819        options will be passed to *delta*, and any **-b**, **-c**, **-e**, **-i**, **-k**, **-l**, **-s** and **-x** options will be  
23820        passed to *get*.

23821        **deledit**    Equivalent to **delget**, except that the *get* phase includes the **-e** option. This option is  
23822        useful for making a checkpoint of the current editing phase. The same options will be  
23823        passed to *delta* as described above, and all the options listed for *get* above except **-e** are  
23824        passed to *edit*.

23825        **diffs**     Write a difference listing between the current version of the files checked out for  
23826        editing and the versions in SCCS format. Any **-r**, **-c**, **-i**, **-x** and **-t** options are passed to  
23827        *get*; any **-l**, **-s**, **-e**, **-f**, **-h** and **-b** options are passed to *diff*. A **-C** option is passed to *diff*  
23828        as **-c**.

23829        **edit**        Equivalent to *get -e*.

23830        **fix**        Remove the named delta, but leave a copy of the delta with the changes that were in it.  
23831        It is useful for fixing small compiler bugs, and so on. It must be followed by a **-r** *SID*  
23832        option. Since **fix** doesn't leave audit trails, it should be used carefully.

23833        **info**        Write a listing of all files being edited. If the **-b** option is given, branches (that is, *SIDs*  
23834        with two or fewer components) are ignored. If a **-u** *user* option is given, then only files  
23835        being edited by the named user are listed. A **-U** option is equivalent to **-u**  
23836        *<current user>*.

23837        **print**       Write out verbose information about the named files, equivalent to *sccs prs*.

23838        **tell**        Write a newline-separated list of the files being edited to standard output. Takes the  
23839        **-b**, **-u** and **-U** options like **info** and **check**.

23840        **unedit**      This is the opposite of an **edit** or a *get -e*. It should be used with caution, since any  
23841        changes made since the *get* will be lost.

23842 **EXIT STATUS**

23843       The following exit values are returned:

23844       0   Successful completion.

23845       &gt;0   An error occurred.

23846 **CONSEQUENCES OF ERRORS**

23847       Default.

23848 **APPLICATION USAGE**

23849       Many of the SCCS utilities take directory names as operands as well as specific filenames. The  
 23850       pseudo-utilities supported by *sccs* are not described as having this capability, but are not  
 23851       prohibited from doing so.

23852 **EXAMPLES**

23853       1. To get a file for editing, edit it and produce a new delta:

23854               *sccs get -e file.c*23855               *ex file.c*23856               *sccs delta file.c*

23857       2. To get a file from another directory:

23858               *sccs -p /usr/src/sccs/s. get cc.c*

23859       or:

23860               *sccs get /usr/src/sccs/s.cc.c*

23861       3. To make a delta of a large number of files in the current directory:

23862               *sccs delta \*.c*

23863       4. To get a list of files being edited that are not on branches:

23864               *sccs info -b*

23865       5. To delta everything being edited by the current user:

23866               *sccs delta \$(sccs tell -U)*

23867       6. In a makefile, to get source files from an SCCS file if it does not already exist:

23868               *SRCS = <list of source files>*23869               *\$(SRCS):*23870               *sccs get \$(REL) \$@*23871 **FUTURE DIRECTIONS**

23872       None.

23873 **SEE ALSO**23874       *admin, delta, get, make, prs, rmdel, sact, unget, val, what.*23875 **CHANGE HISTORY**

23876       First released in Issue 4.

23877 **NAME**23878        *sed* — stream editor23879 **SYNOPSIS**23880        *sed* [-n] *script*[*file*...]23881        *sed* [-n][-e *script*...]...[-f *script\_file*]...[*file*...]23882 **DESCRIPTION**

23883        The *sed* utility is a stream editor that reads one or more text files, makes editing changes  
 23884        according to a script of editing commands, and writes the results to standard output. The script  
 23885        is obtained from either the *script* operand string or a combination of the option-arguments from  
 23886        the -e *script* and -f *script\_file* options.

23887 **OPTIONS**

23888        The *sed* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 23889        that the order of presentation of the -e and -f options is significant.

23890        The following options are supported:

23891        -e *script*

23892            Add the editing commands specified by the *script* option-argument to the end of the  
 23893            script of editing commands. The *script* option-argument has the same properties as the  
 23894            *script* operand, described in the OPERANDS section.

23895        -f *script\_file*

23896            Add the editing commands in the file *script\_file* to the end of the script.

23897        -n        Suppress the default output (in which each line, after it is examined for editing, is  
 23898            written to standard output). Only lines explicitly selected for output will be written.

23899        Multiple -e and -f options may be specified. All commands are added to the script in the order  
 23900        specified, regardless of their origin.

23901 **OPERANDS**

23902        The following operands are supported:

23903        *file*        A pathname of a file whose contents will be read and edited. If multiple *file* operands  
 23904            are specified, the named files will be read in the order specified and the concatenation  
 23905            will be edited. If no *file* operands are specified, the standard input will be used.

23906        *script*      A string to be used as the script of editing commands. The application must not  
 23907            present a *script* that violates the restrictions of a text file except that the final character  
 23908            need not be a newline character.

23909 **STDIN**

23910        The standard input will be used only if no *file* operands are specified. See the INPUT FILES  
 23911        section.

23912 **INPUT FILES**

23913        The input files must be text files. The *script\_files* named by the -f option will consist of editing  
 23914        commands, one per line.

23915 **ENVIRONMENT VARIABLES**

23916        The following environment variables affect the execution of *sed*:

23917        *LANG*        Provide a default value for the internationalisation variables that are unset or null. If  
 23918            *LANG* is unset or null, the corresponding value from the implementation-dependent  
 23919            default locale will be used. If any of the internationalisation variables contains an  
 23920            invalid setting, the utility will behave as if none of the variables had been defined.

23921 **LC\_ALL**  
 23922 If set to a non-empty string value, override the values of all the other  
 23923 internationalisation variables.

23924 **LC\_COLLATE**  
 23925 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 23926 character collating elements within regular expressions.

23927 **LC\_CTYPE**  
 23928 Determine the locale for the interpretation of sequences of bytes of text data as  
 23929 characters (for example, single- versus multi-byte characters in arguments and input  
 23930 files), and the behaviour of character classes within regular expressions.

23931 **LC\_MESSAGES**  
 23932 Determine the locale that should be used to affect the format and contents of diagnostic  
 23933 messages written to standard error.

23934 EX **NLSPATH**  
 23935 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

23936 **ASYNCHRONOUS EVENTS**  
 23937 Default.

23938 **STDOUT**  
 23939 The input files are written to standard output, with the editing commands specified in the script  
 23940 applied. If the *-n* option is specified, only those input lines selected by the script will be written  
 23941 to standard output.

23942 **STDERR**  
 23943 Used only for diagnostic messages.

23944 **OUTPUT FILES**  
 23945 The output files are text files whose formats are dependent on the editing commands given.

23946 **EXTENDED DESCRIPTION**  
 23947 The *script* consists of editing commands, one per line, of the following form:  
 23948 `[address[ , address ]]command[ arguments ]`  
 23949 Zero or more blank characters are accepted before the first address and before *command*.

23950 In default operation, *sed* cyclically copies a line of input, less its terminating newline character,  
 23951 into a *pattern space* (unless there is something left after a **D** command), applies in sequence all  
 23952 commands whose addresses select that pattern space, and at the end of the script copies the  
 23953 pattern space to standard output (except when *-n* is specified) and deletes the pattern space.  
 23954 Whenever the pattern space is written to standard output or a named file, *sed* will immediately  
 23955 follow it with a newline character.

23956 Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent  
 23957 retrieval. The *pattern* and *hold spaces* will each be able to hold at least 8192 bytes.



## 23958 Addresses in sed

23959 An address is either empty, a decimal number that counts input lines cumulatively across files, a  
 23960 "\$" character that addresses the last line of input, or a context address (which consists of a  
 23961 regular expression as described in **Regular Expressions in sed**, preceded and followed by a  
 23962 delimiter, usually a slash).

23963 A command line with no addresses selects every pattern space.

23964 A command line with one address selects each pattern space that matches the address.

23965 A command line with two addresses selects the inclusive range from the first pattern space that  
 23966 matches the first address to the next pattern space that matches the second. (If the second  
 23967 address is a number less than or equal to the line number first selected, only one line will be  
 23968 selected.) Starting at the first line following the selected range, *sed* looks again for the first  
 23969 address. Thereafter the process is repeated.

23970 Editing commands can be applied only to non-selected pattern spaces by use of the negation  
 23971 command "!" (see **Editing Commands in sed**).

## 23972 Regular Expressions in sed

23973 The *sed* utility supports the basic regular expressions described in the **XBD** specification, **Section**  
 23974 **7.3, Basic Regular Expressions**, with the following additions:

- 23975 • In a context address, the construction `\cREc`, where *c* is any character other than a backslash  
 23976 or newline character, is identical to `/RE/`. If the character designated by *c* appears following  
 23977 a backslash, then it is considered to be that literal character, which does not terminate the RE.  
 23978 For example, in the context address `\xabc\xdefx`, the second *x* stands for itself, so that the  
 23979 regular expression is `abcxdef`.
- 23980 • The escape sequence `\n` matches a newline character embedded in the pattern space. A  
 23981 literal newline character must not be used in the regular expression of a context address or in  
 23982 the substitute command.

## 23983 Editing Commands in sed

23984 In the following list of commands, the maximum number of permissible addresses for each  
 23985 command is indicated by `[0addr]`, `[1addr]` or `[2addr]`, representing zero, one or two  
 23986 addresses.

23987 The argument *text* consists of one or more lines. Each embedded newline character in the text  
 23988 must be preceded by a backslash. Other backslashes in text are removed and the following  
 23989 character is treated literally.

23990 The **r** and **w** commands take an optional *rfile* (or *wfile*) parameter, separated from the command  
 23991 letter by one or more blank characters; implementations may allow zero separation as an  
 23992 extension.

23993 The argument *rfile* or the argument *wfile* terminates the command line. Each *wfile* will be created  
 23994 EX before processing begins. Implementations support at least ten *wfile* arguments in the script; the  
 23995 EX actual number (greater than or equal to 10) that will be supported by the implementation is  
 23996 unspecified. The use of the *wfile* parameter causes that file to be initially created, if it does not  
 23997 exist, or will replace the contents of an existing file.

23998 The **b**, **r**, **s**, **t**, **w**, **y**, **!** and **:** commands accept additional arguments. The following synopses  
 23999 indicate which arguments must be separated from the commands by a single space character.

24000 Two of the commands take a *command-list*, which is a list of *sed* commands separated by newline  
 24001 characters, as follows:

```
24002 { command
24003 command
24004 . . .
24005 }
```

24006 The "{" can be preceded with blank characters and can be followed with white space. The  
 24007 *commands* can be preceded by white space. The terminating "}" must be preceded by a newline  
 24008 character and then zero or more blank characters.

24009 [ *2addr* ] { *command-list*  
 24010 } Execute *command-list* only when the pattern space is selected.

24011 [ *1addr* ] a\  
 24012 *text* Write *text* to standard output just before each attempt to fetch a line of input, whether  
 24013 by executing the N command or by beginning a new cycle.

24014 [ *2addr* ] b [ *label* ]  
 24015 Branch to the : command bearing the *label*. If *label* is not specified, branch to the end of  
 24016 the script. The implementation supports *labels* recognised as unique up to at least 8  
 24017 characters; the actual length (greater than or equal to 8) that is supported by the  
 24018 implementation is unspecified. It is unspecified whether exceeding a label length  
 24019 causes an error or a silent truncation.

24020 [ *2addr* ] c\  
 24021 *text* Delete the pattern space. With a 0 or 1 address or at the end of a 2-address range, place  
 24022 *text* on the output.

24023 [ *2addr* ] d  
 24024 Delete the pattern space and start the next cycle.

24025 [ *2addr* ] D  
 24026 Delete the initial segment of the pattern space up to and including the first newline  
 24027 character and start the next cycle.

24028 [ *2addr* ] g  
 24029 Replace the contents of the pattern space by the contents of the hold space.

24030 [ *2addr* ] G  
 24031 Append to the pattern space a newline character followed by the contents of the hold  
 24032 space.

24033 [ *2addr* ] h  
 24034 Replace the contents of the hold space with the contents of the pattern space.

24035 [ *2addr* ] H  
 24036 Append to the hold space a newline character followed by the contents of the pattern  
 24037 space.

24038 [ *1addr* ] i\  
 24039 *text* Write *text* to standard output.

24040 [ *2addr* ] l  
 24041 (The letter ell.) Write the pattern space to standard output in a visually unambiguous  
 24042 form. The characters listed in the table in the **XBD** specification, **Chapter 3, File Format**  
 24043 **Notation** (\, \a, \b, \f, \r, \t, \v) will be written as the corresponding escape  
 24044 sequence; the \n in that table is not applicable. Non-printable characters not in that

24045 table will be written as one three-digit octal number (with a preceding backslash  
 24046 character) for each byte in the character (most significant byte first). If the size of a byte  
 24047 on the system is greater than nine bits, the format used for non-printable characters is  
 24048 implementation-dependent.

24049 Long lines will be folded, with the point of folding indicated by writing a backslash  
 24050 followed by a newline character; the length at which folding occurs is unspecified, but  
 24051 should be appropriate for the output device. The end of each line will be marked with  
 24052 a "\$".

24053 [ 2addr ]n  
 24054 Write the pattern space to standard output if the default output has not been  
 24055 suppressed, and replace the pattern space with the next line of input.

24056 [ 2addr ]N  
 24057 Append the next line of input to the pattern space, using an embedded newline  
 24058 character to separate the appended material from the original material. Note that the  
 24059 current line number changes.

24060 [ 2addr ]p  
 24061 Write the pattern space to standard output.

24062 [ 2addr ]P  
 24063 Write the pattern space, up to the first newline character, to standard output.

24064 [ 1addr ]q  
 24065 Branch to the end of the script and quit without starting a new cycle.

24066 [ 1addr ]r rfile  
 24067 Copy the contents of *rfile* to standard output just before each attempt to fetch a line of  
 24068 input. If *rfile* does not exist or cannot be read, it is treated as if it were an empty file,  
 24069 causing no error condition.

24070 [ 2addr ]s/regular expression/replacement/flags  
 24071 Substitute the *replacement* string for instances of the *regular expression* in the pattern  
 24072 space. Any character other than backslash or newline can be used instead of a slash to  
 24073 delimit the RE and the replacement. Within the RE and the replacement, the RE  
 24074 delimiter itself can be used as a literal character if it is preceded by a backslash.

24075 An ampersand (&) appearing in the *replacement* will be replaced by the string matching  
 24076 the RE. The special meaning of "&" in this context can be suppressed by preceding it by  
 24077 backslash. The characters \n, where *n* is a digit, will be replaced by the text matched by  
 24078 the corresponding backreference expression. For each backslash (\) encountered in  
 24079 scanning *replacement* from beginning to end, the following character loses its special  
 24080 meaning (if any). It is unspecified what special meaning is given to any character other  
 24081 than &, \ or digits.

24082 A line can be split by substituting a newline character into it. The application must  
 24083 escape the newline character in the *replacement* by preceding it by backslash. A  
 24084 substitution is considered to have been performed even if the replacement string is  
 24085 identical to the string that it replaces.

24086 The value of *flags* must be zero or more of:

24087 *n* Substitute for the *n*th occurrence only of the *regular expression* found within  
 24088 the pattern space.

24089 *g* Globally substitute for all non-overlapping instances of the *regular expression*  
 24090 rather than just the first one. If both *g* and *n* are specified, the results are

24091 unspecified.

24092 **p** Write the pattern space to standard output if a replacement was made.

24093 **w wfile** Write. Append the pattern space to *wfile* if a replacement was made.

24094 **[ 2addr ]t [ label ]**

24095 Test. Branch to the **:** command bearing the *label* if any substitutions have been made

24096 since the most recent reading of an input line or execution of a **t**. If *label* is not specified,

24097 branch to the end of the script.

24098 **[ 2addr ]w wfile**

24099 Append (write) the pattern space to *wfile*.

24100 **[ 2addr ]x**

24101 Exchange the contents of the pattern and hold spaces.

24102 **[ 2addr ]y/string1/string2/**

24103 Replace all occurrences of characters in *string1* with the corresponding characters in

24104 *string2*. If the number of characters in *string1* and *string2* are not equal, or if any of the

24105 characters in *string1* appear more than once, the results are undefined. Any character

24106 other than backslash or newline can be used instead of slash to delimit the strings.

24107 Within *string1* and *string2*, the delimiter itself can be used as a literal character if it is

24108 preceded by a backslash.

24109 **[ 2addr ]!command**

24110 **[ 2addr ]!{command-list**

24111 **}** Apply the *command* or *command-list* only to the lines that are not selected by the

24112 addresses.

24113 **[ 0addr ]:label**

24114 This command does nothing; it bears a *label* for the **b** and **t** commands to branch to.

24115 **[ 1addr ]=**

24116 Write the following to standard output:

24117 `"%d\n", <current line number>`

24118 **[ 0addr ]** An empty command is ignored.

24119 **[ 0addr ]#**

24120 The `"#"` and the remainder of the line are ignored (treated as a comment), with the

24121 single exception that if the first two characters in the file are `#n`, the default output is

24122 suppressed; this is the equivalent of specifying `-n` on the command line.

24123 **EXIT STATUS**

24124 The following exit values are returned:

24125 **0** Successful completion.

24126 **>0** An error occurred.

24127 **CONSEQUENCES OF ERRORS**

24128 Default.

24129 **APPLICATION USAGE**

24130 Regular expressions match entire strings, not just individual lines, but a newline character is

24131 matched by `\n` in a *sed* RE; a newline character is not allowed in an RE. Also note that `\n` cannot

24132 be used to match a newline character at the end of an arbitrary input line; newline characters

24133 appear in the pattern space as a result of the **N** editing command.

24134 **EXAMPLES**

24135 This *sed* script simulates the BSD *cat -s* command, squeezing excess blank lines from standard  
 24136 input.

```

24137 sed -n '
24138 # Write non-empty lines.
24139 ./ {
24140 p
24141 d
24142 }
24143 # Write a single empty line, then look for more empty lines.
24144 /^$/ p
24145 # Get next line, discard the held <newline> (empty line),
24146 # and look for more empty lines.
24147 :Empty
24148 /^$/ {
24149 N
24150 s/./ /
24151 b Empty
24152 }
24153 # Write the non-empty line before going back to search
24154 # for the first in a set of empty lines.
24155 p
24156 '

```

24157 **FUTURE DIRECTIONS**

24158 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 24159 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 24160 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 24161 finalised.

24162 **SEE ALSO**

24163 *awk, ed, grep.*

24164 **CHANGE HISTORY**

24165 First released in Issue 2.

24166 **Issue 4**

24167 Aligned with the ISO/IEC 9945-2: 1993 standard.

24168 **Issue 5**

24169 FUTURE DIRECTIONS section added.

## 24170 NAME

24171 sh — shell, the standard command language interpreter

## 24172 SYNOPSIS

24173 EX sh [-abCefimnuvx][-o option][+abCefmnuvx][+o option]  
24174 [command\_file [argument...]]

24175 EX sh [-abCefimnuvx][-o option][+abCefmnuvx][+o option]command\_string  
24176 [command\_name [argument...]]

24177 EX sh -s[-abCefimnuvx][-o option][+abCefmnuvx][+o option][argument]

## 24178 DESCRIPTION

24179 The *sh* utility is a command language interpreter that executes commands read from a  
24180 command-line string, the standard input or a specified file. The commands to be executed must  
24181 be expressed in the language described in Chapter 2 on page 19.

24182 EX Pathname expansion will not fail due to the size of a file.

24183 Shell input and output redirections will have an implementation-dependent offset maximum  
24184 that will be established in the open file description.

## 24185 OPTIONS

24186 The *sh* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

24187 The **-a**, **-b**, **-C**, **-e**, **-f**, **-m**, **-n**, **-o option**, **-u**, **-v** and **-x** options are described as part of the *set*  
24188 EX utility in Section 2.14 on page 67. The option letters derived from the *set* special built-in are also  
24189 accepted with a leading plus sign (+) instead of a leading hyphen (meaning the reverse case of  
24190 the option as described in this specification).

24191 The following additional options are supported:

24192 **-c** Read commands from the *command\_string* operand. Set the value of special parameter  
24193 0 (see Section 2.5.2 on page 27) from the value of the *command\_name* operand and the  
24194 positional parameters (\$1, \$2, and so on) in sequence from the remaining *argument*  
24195 operands. No commands will be read from the standard input.

24196 **-i** Specify that the shell is *interactive*; see below. An implementation may treat specifying  
24197 the **-i** option as an error if the real user ID of the calling process does not equal the  
24198 effective user ID or if the real group ID does not equal the effective user ID.

24199 **-s** Read commands from the standard input.

24200 If there are no operands and the **-c** option is not specified, the **-s** option is assumed.

24201 If the **-i** option is present, or if there are no operands and the shell's standard input and standard  
24202 error are attached to a terminal, the shell is considered to be *interactive*.

## 24203 OPERANDS

24204 The following operands are supported:

24205 **-** A single hyphen is treated as the first operand and then ignored. If both "-" and "--" are  
24206 given as arguments, or if other operands precede the single hyphen, the results are  
24207 undefined.

24208 *argument*

24209 The positional parameters (\$1, \$2 and so on) will be set to *arguments*, if any.

24210 *command\_file*

24211 The pathname of a file containing commands. If the pathname contains one or more  
24212 slash characters, the implementation will attempt to read that file; the file need not be

24213 executable. If the pathname does not contain a slash character:

24214 • The implementation will attempt to read that file from the current working

24215 directory; the file need not be executable.

24216 • If the file is not in the current working directory, the implementation may perform a

24217 search for an executable file using the value of *PATH*, as described in **Command**

24218 **Search and Execution** on page 47.

24219 Special parameter 0 (see Section 2.5.2 on page 27) is set to the value of *command\_file*. If

24220 *sh* is called using a synopsis form that omits *command\_file*, special parameter 0 is set to

24221 the value of the first argument passed to *sh* from its parent (for example, *argv*[0] for a C

24222 program), which is normally a pathname used to execute the *sh* utility.

24223 *command\_name*

24224 A string assigned to special parameter 0 when executing the commands in

24225 *command\_string*. If *command\_name* is not specified, special parameter 0 is set to the

24226 value of the first argument passed to *sh* from its parent (for example, *argv*[0] for a C

24227 program), which is normally a pathname used to execute the *sh* utility.

24228 *command\_string*

24229 A string that is interpreted by the shell as one or more commands, as if the string were

24230 the argument to the **XSH** specification *system*() function. If the *command\_string*

24231 operand is an empty string, *sh* will exit with a zero exit status.

24232 **STDIN**

24233 The standard input will be used only if one of the following is true:

24234 • The **-s** option is specified.

24235 • The **-c** option is not specified and no operands are specified.

24236 • The script executes one or more commands that require input from standard input (such as a

24237 *read* command that does not redirect its input).

24238 See the INPUT FILES section.

24239 When the shell is using standard input and it invokes a command that also uses standard input,

24240 the shell ensures that the standard input file pointer points directly after the command it has

24241 read when the command begins execution. It will not read ahead in such a manner that any

24242 characters intended to be read by the invoked command are consumed by the shell (whether

24243 interpreted by the shell or not) or that characters that are not read by the invoked command are

24244 not seen by the shell. When the command expecting to read standard input is started

24245 asynchronously by an interactive shell, it is unspecified whether characters are read by the

24246 command or interpreted by the shell.

24247 If the standard input to *sh* is a FIFO or terminal device and is set to non-blocking reads, then *sh*

24248 will enable blocking reads on standard input. This will remain in effect when the command

24249 completes. (This concerns an instance of *sh* that has been invoked, probably by a C-language

24250 program, with standard input that has been opened using the *O\_NONBLOCK* flag; see *open*() in

24251 the **XSH** specification. If the shell did not reset this flag, it would immediately terminate

24252 because no input data would be available yet and that would be considered the same as end-of-

24253 file.)

24254 **INPUT FILES**

24255 The input file must be a text file, except that line lengths are unlimited. If the input file is empty

24256 or consists solely of blank lines or comments, or both, *sh* will exit with a zero exit status.

24257 **ENVIRONMENT VARIABLES**

24258       The following environment variables affect the execution of *sh*:

24259       **FCEDIT**

24260           This variable, when expanded by the shell, determines the default value for the  
24261           **-e** *editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* will be used as  
24262           the editor.

24263       **HISTFILE**

24264           Determine a pathname naming a command history file. If the *HISTFILE* variable is not  
24265           set, the shell may attempt to access or create a file **.sh\_history** in the user's home  
24266           directory. If the shell cannot obtain both read and write access to, or create, the history  
24267           file, it will use an unspecified mechanism that allows the history to operate properly.  
24268           (References to history "file" in this section are understood to mean this unspecified  
24269           mechanism in such cases.) An implementation may choose to access this variable only  
24270           when initialising the history file; this initialisation will occur when *fc* or *sh* first attempt  
24271           to retrieve entries from, or add entries to, the file, as the result of commands issued by  
24272           the user, the file named by the *ENV* variable, or implementation-dependent system  
24273           startup files. (The initialisation process for the history file can be dependent on the  
24274           system startup files, in that they may contain commands that will effectively preempt  
24275           the user's settings of *HISTFILE* and *HISTSIZE*. For example, function definition  
24276           commands are recorded in the history file, unless the *set -o nolog* option is set. If the  
24277           system administrator includes function definitions in some system startup file called  
24278           before the *ENV* file, the history file will be initialised before the user gets a chance to  
24279           influence its characteristics.) In some historical shells, the history file is initialised just  
24280           after the *ENV* file has been processed. Therefore, it is implementation-dependent  
24281           whether changes made to *HISTFILE* after the history file has been initialised are  
24282           effective. Implementations may choose to disable the history list mechanism for users  
24283           with appropriate privileges who do not set *HISTFILE*; the specific circumstances under  
24284           which this will occur are implementation-dependent. If more than one instance of the  
24285           shell is using the same history file, it is unspecified how updates to the history file from  
24286           those shells interact. As entries are deleted from the history file, they will be deleted  
24287           oldest first. It is unspecified when history file entries are physically removed from the  
24288           history file.

24289       **HISTSIZE**

24290           Determine a decimal number representing the limit to the number of previous  
24291           commands that are accessible. If this variable is unset, an unspecified default greater  
24292           than or equal to 128 will be used. The maximum number of commands in the history  
24293           list is unspecified, but will be at least 128. An implementation may choose to access  
24294           this variable only when initialising the history file, as described under *HISTFILE*.  
24295           Therefore, it is unspecified whether changes made to *HISTSIZE* after the history file has  
24296           been initialised are effective.

24297       **HOME**   Determine the pathname of the user's home directory. The contents of *HOME* are used  
24298           in Tilde Expansion as described in Section 2.6.1 on page 32.

24299       **IFS**     **Input field separators:** a string treated as a list of characters that is used for field splitting  
24300           and to split lines into words with the *read* command. See Section 2.6.5 on page 38. If  
24301           **IFS** is not set, the shell behaves as if the value of *IFS* were the space, tab and newline  
24302           characters. Implementations may ignore the value of *IFS* in the environment at the  
24303           time *sh* is invoked, treating *IFS* as if it were not set.

24304       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
24305           **LANG** is unset or null, the corresponding value from the implementation-dependent



|          |                            |                                                                                                       |
|----------|----------------------------|-------------------------------------------------------------------------------------------------------|
| 24306    |                            | default locale will be used. If any of the internationalisation variables contains an                 |
| 24307    |                            | invalid setting, the utility will behave as if none of the variables had been defined.                |
| 24308    | <i>LC_ALL</i>              |                                                                                                       |
| 24309    |                            | If set to a non-empty string value, override the values of all the other                              |
| 24310    |                            | internationalisation variables.                                                                       |
| 24311    | <i>LC_COLLATE</i>          |                                                                                                       |
| 24312    |                            | Determine the behaviour of range expressions, equivalence classes and multi-character                 |
| 24313    |                            | collating elements within pattern matching.                                                           |
| 24314    | <i>LC_CTYPE</i>            |                                                                                                       |
| 24315    |                            | Determine the locale for the interpretation of sequences of bytes of text data as                     |
| 24316    |                            | characters (for example, single- versus multi-byte characters in arguments and input                  |
| 24317    |                            | files), which characters are defined as letters (character class <b>alpha</b> ), and the behaviour    |
| 24318    |                            | of character classes within pattern matching.                                                         |
| 24319    | <i>LC_MESSAGES</i>         |                                                                                                       |
| 24320    |                            | Determine the locale that should be used to affect the format and contents of diagnostic              |
| 24321    |                            | messages written to standard error.                                                                   |
| 24322    | <i>MAIL</i>                | Determine a pathname of the user's mailbox file for purposes of incoming mail                         |
| 24323    |                            | notification. If this variable is set, the shell will inform the user if the file named by the        |
| 24324    |                            | variable is created or if its modification time has changed. Informing the user is                    |
| 24325    |                            | accomplished by writing a string of unspecified format to standard error prior to the                 |
| 24326    |                            | writing of the next primary prompt string after the completion of an interval defined by              |
| 24327    |                            | the <i>MAILCHECK</i> variable. The user will be informed only if <i>MAIL</i> is set and               |
| 24328    |                            | <i>MAILPATH</i> is not set.                                                                           |
| 24329    | <i>MAILCHECK</i>           |                                                                                                       |
| 24330    |                            | Establish a decimal integer value that specifies how often (in seconds) the shell will                |
| 24331    |                            | check for the arrival of mail in the files specified by the <i>MAILPATH</i> or <i>MAIL</i> variables. |
| 24332    |                            | The default value is 600 seconds. If set to zero, the shell will check before issuing each            |
| 24333    |                            | primary prompt.                                                                                       |
| 24334    | <i>MAILPATH</i>            |                                                                                                       |
| 24335    |                            | Provide a list of pathnames and optional messages separated by colons. If this variable               |
| 24336    |                            | is set, the shell will inform the user if any of the files named by the variable are created          |
| 24337    |                            | or if any of their modification times change. (See the preceding entry for <i>MAIL</i> for            |
| 24338    |                            | descriptions of mail arrival and user informing.) Each pathname can be followed by                    |
| 24339    |                            | "%" and a string that will be subjected to parameter expansion and written to standard                |
| 24340    |                            | error when the modification time changes. If a "%" character in the pathname is                       |
| 24341    |                            | preceded by a backslash, it will be treated as a literal "%" in the pathname. The default             |
| 24342    |                            | message is unspecified.                                                                               |
| 24343    |                            | The <i>MAILPATH</i> environment variable takes precedence over the <i>MAIL</i> variable.              |
| 24344 EX | <i>NLSPATH</i>             |                                                                                                       |
| 24345    |                            | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .               |
| 24346    | <i>PATH</i>                | Establish a string formatted as described in the <b>XBD</b> specification, <b>Chapter 6</b> ,         |
| 24347    |                            | <b>Environment Variables</b> , used to effect command interpretation. See <b>Command Search</b>       |
| 24348    |                            | <b>and Execution</b> on page 47.                                                                      |
| 24349    | <b>ASYNCHRONOUS EVENTS</b> |                                                                                                       |
| 24350    |                            | Default.                                                                                              |

24351 **STDOUT**

24352 See the STDERR section.

24353 **STDERR**

24354 Except as otherwise stated (by the descriptions of any invoked utilities or in interactive mode),  
 24355 standard error is used only for diagnostic messages.

24356 **OUTPUT FILES**

24357 None.

24358 **EXTENDED DESCRIPTION**

24359 See Chapter 2. The following additional capabilities are supported.

24360 **Command History List**

24361 When the *sh* utility is being used interactively, it maintains a list of commands previously  
 24362 entered from the terminal in the file named by the *HISTFILE* environment variable. The type,  
 24363 size and internal format of this file are unspecified. Multiple *sh* processes can share access to the  
 24364 file for a user, if file access permissions allow this; see the description of the *HISTFILE*  
 24365 environment variable.

24366 **Command Line Editing**

24367 When *sh* is being used interactively from a terminal, the current command and the command  
 24368 history (see *fc*) can be edited using *vi-mode* command line editing. This mode uses commands,  
 24369 described below, similar to a subset of those described in the *vi* utility. Implementations may  
 24370 offer other command line editing modes corresponding to other editing utilities.

24371 The command *set -o v* enables vi-mode editing and places *sh* into *vi* insert mode (see **Command**  
 24372 **Line Editing (vi-mode)**). This command also disables any other editing mode that the  
 24373 implementation may provide. The command *set +o v* disables vi-mode editing.

24374 Certain block-mode terminals may be unable to support shell command line editing. If a  
 24375 terminal is unable to provide either edit mode, it need not be possible to *set -o v* when using the  
 24376 shell on this terminal.

24377 In the following sections, the characters erase, interrupt, kill and end-of-file are those set by the  
 24378 *stty* utility.

24379 **Command Line Editing (vi-mode)**24380 With vi-mode enabled, *sh* can be switched between insert mode and command mode.

24381 When in insert mode, an entered character will be inserted into the command line, except as  
 24382 noted in **vi Line Editing Insert Mode** on page 659. Upon entering *sh* and after termination of the  
 24383 previous command, *sh* will be in insert mode.

24384 Typing an escape character will switch *sh* into command mode (see **vi Line Editing Command**  
 24385 **Mode** on page 659). In command mode, an entered character will either invoke a defined  
 24386 operation, be used as part of a multi-character operation or be treated as an error. A character  
 24387 that is not recognised as part of an editing command will terminate any specific editing  
 24388 command and will alert the terminal. Typing the *interrupt* character in command mode will  
 24389 cause *sh* to terminate command line editing on the current command line, reissue the prompt on  
 24390 the next line of the terminal and reset the command history (see *fc*) so that the most recently  
 24391 executed command is the previous command (that is, the command that was being edited when  
 24392 it was interrupted is not reentered into the history).

24393 In the following sections, the phrase “move the cursor to the beginning of the word” means  
 24394 “move the cursor to the first character of the current word” and the phrase “move the cursor to  
 24395 the end of the word” means “move the cursor to the last character of the current word”. The  
 24396 phrase “beginning of the command line” indicates the point between the end of the prompt  
 24397 string issued by the shell (or the beginning of the terminal line, if there is no prompt string) and  
 24398 the first character of the command text.

### 24399 **vi Line Editing Insert Mode**

24400 While in insert mode, any character typed will be inserted in the current command line, unless it  
 24401 is from the following set.

24402 newline

24403 Execute the current command line being edited.

24404 *erase* Delete the character previous to the current cursor position and move the current  
 24405 cursor position back one character. In insert mode, characters will be erased from both  
 24406 the screen and the buffer when backspacing.

24407 *interrupt*

24408 Terminate command line editing with the same effects as described for interrupting  
 24409 command mode; see **Command Line Editing (vi-mode)** on page 658.

24410 *kill* Clear all the characters from the input line.

24411 <control>-V

24412 Insert the next character input, even if the character is otherwise a special insert mode  
 24413 character.

24414 <control>-W

24415 Delete the characters from the one preceding the cursor to the preceding word  
 24416 boundary. The word boundary in this case is the closer to the cursor of either the  
 24417 beginning of the line or a character that is in neither the **blank** nor **punct** character  
 24418 classification of the current locale.

24419 \ On some systems, when a backslash is followed by an *erase* or *kill* character, that  
 24420 character will be inserted into the input line. This is not actually a feature of *sh*  
 24421 command line editing insert mode, but one of terminal line drivers when the *stty iexten*  
 24422 flag is set. Otherwise, the backslash itself will be inserted into the input line.

24423 *end-of-file*

24424 Interpreted as the end of input in *sh*. This interpretation will occur only at the  
 24425 beginning of an input line. If end-of-file is entered other than at the beginning of the  
 24426 line, the results are unspecified.

24427 <ESC> Place *sh* into command mode.

### 24428 **vi Line Editing Command Mode**

24429 In command mode for the command line editing feature, decimal digits not beginning with 0  
 24430 that precede a command letter will be remembered. Some commands use these decimal digits  
 24431 as a count number that affects the operation.

24432 The term *motion command* represents one of the commands:

24433 <space> 0 b F l W ^ \$ ; E f T w | , B e h t

24434 Any command that modifies the current line will cause a copy of the current line to be made at  
 24435 the end of the command history, the current line will become that copy, and the edit will be

24436 performed on that copy.

24437 Any command that is preceded by *count* will take a count (the numeric value of any preceding  
 24438 decimal digits). Unless otherwise noted, this count will cause the specified operation to repeat  
 24439 by the number of times specified by the count. Also unless otherwise noted, a *count* that is out of  
 24440 range is considered an error condition and will alert the terminal, but neither the cursor position,  
 24441 nor the command line, will change.

24442 The terms *word* and *bigword* are used as defined in the *vi* description. The term *save buffer*  
 24443 corresponds to the term *unnamed buffer* in *vi*.

24444 The following commands are recognised in command mode:

24445 *newline*

24446 Execute the current command line being edited.

24447 <control>-L

24448 Redraw the current command line. Position the cursor at the same location on the new  
 24449 command line.

24450 # Insert the character # at the beginning of the current command line and treat the  
 24451 current command line as a comment. This line will be entered into the command  
 24452 history; see *fc*.

24453 = Display the possible shell word expansions (see Section 2.6 on page 31) of the bigword  
 24454 at the current command line position. These expansions will be displayed on  
 24455 subsequent terminal lines. If the bigword contains none of the characters "?", "\*" or "[",  
 24456 an asterisk (\*) will be implicitly assumed at the end. If any directories are matched,  
 24457 these expansions will have a "/" character appended. After the expansion, the line will  
 24458 be redrawn, the cursor will be repositioned at the current cursor position, and *sh* will be  
 24459 placed in command mode.

24460 \ Perform pathname expansion (see Section 2.6.6 on page 39) on the current bigword, up  
 24461 to the largest set of characters that can be matched uniquely. If the bigword contains  
 24462 none of the characters "?", "\*" or "[", an asterisk (\*) will be implicitly assumed at the end.  
 24463 This maximal expansion then will replace the original bigword in the command line,  
 24464 and the cursor will be placed after this expansion. If the resulting bigword completely  
 24465 and uniquely matches a directory, a "/" character will be inserted directly after the  
 24466 bigword. If some other file is completely matched, a single space character will be  
 24467 inserted after the bigword. After this operation, *sh* will be placed in insert mode.

24468 \* Perform pathname expansion on the current bigword and insert all expansions into the  
 24469 command to replace the current bigword, with each expansion separated by a single  
 24470 space character. If at the end of the line, the current cursor position will be moved to  
 24471 the first column position following the expansions and *sh* will be placed in insert mode.  
 24472 Otherwise, the current cursor position will be the last column position of the first  
 24473 character after the expansions and *sh* will be placed in insert mode. If the current  
 24474 bigword contains none of the characters "?", "\*" or "[", before the operation, an asterisk  
 24475 will be implicitly assumed at the end.

24476 @*letter*

24477 Insert the value of the alias named *\_letter*. The symbol *letter* represents a single  
 24478 alphabetic character from the portable character set; implementations may support  
 24479 additional characters as an extension. If the alias *\_letter* contains other editing  
 24480 commands, these commands will be performed as part of the insertion. If no alias  
 24481 *\_letter* is enabled, this command will have no effect.

24482        **[count]~**  
 24483            Convert, if the current character is a lower-case letter, to the equivalent upper-case  
 24484            letter and *viceversa*, as prescribed by the current locale. The current cursor position  
 24485            then will be advanced by one character. If the cursor was positioned on the last  
 24486            character of the line, the case conversion will occur, but the cursor will not advance. If  
 24487            the ~ command is preceded by a *count*, that number of characters will be converted, and  
 24488            the cursor will be advanced to the character position after the last character converted.  
 24489            If the *count* is larger than the number of characters after the cursor, this is not  
 24490            considered an error; the cursor will advance to the last character on the line.

24491        **[count].**  
 24492            Repeat the most recent non-motion command, even if it was executed on an earlier  
 24493            command line. If the previous command was preceded by a *count*, and no count is  
 24494            given on the "." command, the count from the previous command will be included as  
 24495            part of the repeated command. If the "." command is preceded by a *count*, this will  
 24496            override any *count* argument to the previous command. The *count* specified in the "."  
 24497            command will become the count for subsequent "." commands issued without a count.

24498        **[number]v**  
 24499            Invoke the *vi* editor to edit the current command line in a temporary file. When the  
 24500            editor exits, the commands in the temporary file will be executed. If a *number* is  
 24501            prefixed to the command, it specifies the command number in the command history to  
 24502            be edited, rather than the current command line.

24503        **[count]l**    (ell)  
 24504        **[count]<space>**  
 24505            Move the current cursor position to the next character position. If the cursor was  
 24506            positioned on the last character of the line, the terminal will be alerted and the cursor  
 24507            will not be advanced. If the *count* is larger than the number of characters after the  
 24508            cursor, this is not considered an error; the cursor will advance to the last character on  
 24509            the line.

24510        **[count]h**  
 24511            Move the current cursor position to the *count*th (default 1) previous character position.  
 24512            If the cursor was positioned on the first character of the line, the terminal will be alerted  
 24513            and the cursor will not be moved. If the count is larger than the number of characters  
 24514            before the cursor, this is not considered an error; the cursor will move to the first  
 24515            character on the line.

24516        **[count]w**  
 24517            Move to the start of the next word. If the cursor was positioned on the last character of  
 24518            the line, the terminal will be alerted and the cursor will not be advanced. If the *count* is  
 24519            larger than the number of words after the cursor, this is not considered an error; the  
 24520            cursor will advance to the last character on the line.

24521        **[count]W**  
 24522            Move to the start of the next bigword. If the cursor was positioned on the last character  
 24523            of the line, the terminal will be alerted and the cursor will not be advanced. If the *count*  
 24524            is larger than the number of bigwords after the cursor, this is not considered an error;  
 24525            the cursor will advance to the last character on the line.

24526        **[count]e**  
 24527            Move to the end of the current word. If at the end of a word, move to the end of the  
 24528            next word. If the cursor was positioned on the last character of the line, the terminal  
 24529            will be alerted and the cursor will not be advanced. If the *count* is larger than the  
 24530            number of words after the cursor, this is not considered an error; the cursor will

|       |                  |                                                                                                   |
|-------|------------------|---------------------------------------------------------------------------------------------------|
| 24531 |                  | advance to the last character on the line.                                                        |
| 24532 | <b>[count]E</b>  |                                                                                                   |
| 24533 |                  | Move to the end of the current bigword. If at the end of a bigword, move to the end of            |
| 24534 |                  | the next bigword. If the cursor was positioned on the last character of the line, the             |
| 24535 |                  | terminal will be alerted and the cursor will not be advanced. If the <i>count</i> is larger than  |
| 24536 |                  | the number of bigwords after the cursor, this is not considered an error; the cursor will         |
| 24537 |                  | advance to the last character on the line.                                                        |
| 24538 | <b>[count]b</b>  |                                                                                                   |
| 24539 |                  | Move to the beginning of the current word. If at the beginning of a word, move to the             |
| 24540 |                  | beginning of the previous word. If the cursor was positioned on the first character of            |
| 24541 |                  | the line, the terminal will be alerted and the cursor will not be moved. If the <i>count</i> is   |
| 24542 |                  | larger than the number of words preceding the cursor, this is not considered an error;            |
| 24543 |                  | the cursor will return to the first character on the line.                                        |
| 24544 | <b>[count]B</b>  |                                                                                                   |
| 24545 |                  | Move to the beginning of the current bigword. If at the beginning of a bigword, move              |
| 24546 |                  | to the beginning of the previous bigword. If the cursor was positioned on the first               |
| 24547 |                  | character of the line, the terminal will be alerted and the cursor will not be moved. If          |
| 24548 |                  | the <i>count</i> is larger than the number of bigwords preceding the cursor, this is not          |
| 24549 |                  | considered an error; the cursor will return to the first character on the line.                   |
| 24550 | <b>^</b>         | Move the current cursor position to the first character on the input line that is not a           |
| 24551 |                  | blank character.                                                                                  |
| 24552 | <b>\$</b>        | Move to the last character position on the current command line.                                  |
| 24553 | <b>0</b>         | (Zero.) Move to the first character position on the current command line.                         |
| 24554 | <b>[count]  </b> |                                                                                                   |
| 24555 |                  | Move to the <i>count</i> th character position on the current command line. If no number is       |
| 24556 |                  | specified, move to the first position. The first character position is numbered 1. If the         |
| 24557 |                  | count is larger than the number of characters on the line, this is not considered an error;       |
| 24558 |                  | the cursor will be placed on the last character on the line.                                      |
| 24559 | <b>[count]fc</b> |                                                                                                   |
| 24560 |                  | Move to the first occurrence of the character <i>c</i> that occurs after the current cursor       |
| 24561 |                  | position. If the cursor was positioned on the last character of the line, the terminal will       |
| 24562 |                  | be alerted and the cursor will not be advanced. If the character <i>c</i> does not occur in the   |
| 24563 |                  | line after the current cursor position, the terminal will be alerted and the cursor will not      |
| 24564 |                  | be moved.                                                                                         |
| 24565 | <b>[count]Fc</b> |                                                                                                   |
| 24566 |                  | Move to the first occurrence of the character <i>c</i> that occurs before the current cursor      |
| 24567 |                  | position. If the cursor was positioned on the first character of the line, the terminal will      |
| 24568 |                  | be alerted and the cursor will not be moved. If the character <i>c</i> does not occur in the line |
| 24569 |                  | before the current cursor position, the terminal will be alerted and the cursor will not be       |
| 24570 |                  | moved.                                                                                            |
| 24571 | <b>[count]tc</b> |                                                                                                   |
| 24572 |                  | Move to the character before the first occurrence of the character <i>c</i> that occurs after the |
| 24573 |                  | current cursor position. If the cursor was positioned on the last character of the line,          |
| 24574 |                  | the terminal will be alerted and the cursor will not be advanced. If the character <i>c</i> does  |
| 24575 |                  | not occur in the line after the current cursor position, the terminal will be alerted and         |
| 24576 |                  | the cursor will not be moved.                                                                     |

24577        **[count]Tc**  
 24578            Move to the character after the first occurrence of the character *c* that occurs before the  
 24579            current cursor position. If the cursor was positioned on the first character of the line,  
 24580            the terminal will be alerted and the cursor will not be moved. If the character *c* does  
 24581            not occur in the line before the current cursor position, the terminal will be alerted and  
 24582            the cursor will not be moved.

24583        **[count];**  
 24584            Repeat the most recent **f**, **F**, **t** or **T** command. Any number argument on that previous  
 24585            command will be ignored. Errors are those described for the repeated command.

24586        **[count],**  
 24587            Repeat the most recent **f**, **F**, **t** or **T** command. Any number argument on that previous  
 24588            command will be ignored. However, reverse the direction of that command.

24589        **a**        Enter insert mode after the current cursor position. Characters that are entered will be  
 24590            inserted before the next character.

24591        **A**        Enter insert mode after the end of the current command line.

24592        **i**        Enter insert mode at the current cursor position. Characters that are entered will be  
 24593            inserted before the current character.

24594        **I**        Enter insert mode at the beginning of the current command line.

24595        **R**        Enter insert mode, replacing characters from the command line beginning at the current  
 24596            cursor position.

24597        **[count]c motion**  
 24598            Delete the characters between the current cursor position and the cursor position that  
 24599            would result from the specified *motion* command. Then enter insert mode before the  
 24600            first character following any deleted characters. If *count* is specified, it will be applied  
 24601            to the motion command. A *count* will be ignored for the following motion commands:

24602            0        ^        \$        c

24603            If the *motion* command is the character **c**, the current command line will be cleared and  
 24604            insert mode will be entered. If the *motion* command would move the current cursor  
 24605            position toward the beginning of the command line, the character under the current  
 24606            cursor position will not be deleted. If the motion command would move the current  
 24607            cursor position toward the end of the command line, the character under the current  
 24608            cursor position will be deleted. If the *count* is larger than the number of characters  
 24609            between the current cursor position and the end of the command line toward which the  
 24610            motion command would move the cursor, this is not considered an error; all of the  
 24611            remaining characters in the aforementioned range will be deleted and insert mode will  
 24612            be entered. If the motion command is invalid, the terminal will be alerted, the cursor  
 24613            will not be moved, and no text will be deleted.

24614        **C**        Delete from the current character to the end of the line and enter insert mode at the new  
 24615            end-of-line.

24616        **S**        Clear the entire current command line and enter insert mode.

24617        **[count]rc**  
 24618            Replace the current character with the character *c*. With a number *count*, replace the  
 24619            current and the following *count*−1 characters. After this command, the current cursor  
 24620            position will be on the last character that was changed. If the *count* is larger than the  
 24621            number of characters after the cursor, this is not considered an error; all of the  
 24622            remaining characters will be changed.

24623 [count]\_  
 24624 Append a space character after the current character position and then append the last  
 24625 bigword in the previous input line after the space character. Then enter insert mode  
 24626 after the last character just appended. With a number *count*, append the *count*th  
 24627 bigword in the previous line.

24628 [count]x  
 24629 Delete the character at the current cursor position and place the deleted characters in  
 24630 the save buffer. If the cursor was positioned on the last character of the line, the  
 24631 character will be deleted and the cursor position will be moved to the previous  
 24632 character (the new last character). If the *count* is larger than the number of characters  
 24633 after the cursor, this is not considered an error; all the characters from the cursor to the  
 24634 end of the line will be deleted.

24635 [count]X  
 24636 Delete the character before the current cursor position and place the deleted characters  
 24637 in the save buffer. The character under the current cursor position will not change. If  
 24638 the cursor was positioned on the first character of the line, the terminal will be alerted,  
 24639 and the X command will have no effect. If the line contained a single character, the X  
 24640 command will have no effect. If the line contained no characters, the terminal will be  
 24641 alerted and the cursor will not be moved. If the *count* is larger than the number of  
 24642 characters before the cursor, this is not considered an error; all the characters from  
 24643 before the cursor to the beginning of the line will be deleted.

24644 [count]d *motion*  
 24645 Delete the characters between the current cursor position and the character position  
 24646 that would result from the *motion* command. A number *count* repeats the *motion*  
 24647 command *count* times. If the motion command would move toward the beginning of  
 24648 the command line, the character under the current cursor position will not be deleted.  
 24649 If the motion command is d, the entire current command line will be cleared. If the  
 24650 *count* is larger than the number of characters between the current cursor position and  
 24651 the end of the command line toward which the motion command would move the  
 24652 cursor, this is not considered an error; all of the remaining characters in the  
 24653 aforementioned range will be deleted. The deleted characters will be placed in the save  
 24654 buffer.

24655 D Delete all characters from the current cursor position to the end of the line. The deleted  
 24656 characters will be placed in the save buffer.

24657 [count]y *motion*  
 24658 Yank (that is, copy) the characters from the current cursor position to the position  
 24659 resulting from the *motion* command into a save buffer. A number *count* will be applied  
 24660 to the *motion* command. If the motion command would move toward the beginning of  
 24661 the command line, the character under the current cursor position will not be included  
 24662 in the set of yanked characters. If the motion command is y, the entire current  
 24663 command line will be yanked into the save buffer. The current cursor position will be  
 24664 unchanged. If the *count* is larger than the number of characters between the current  
 24665 cursor position and the end of the command line toward which the motion command  
 24666 would move the cursor, this is not considered an error; all of the remaining characters  
 24667 in the aforementioned range will be yanked.

24668 Y Yank the characters from the current cursor position to the end of the line into the save  
 24669 buffer. The current character position will be unchanged.

24670 [count]p  
 24671 Put a copy of the current contents of the save buffer after the current cursor position.



|       |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24672 |                               | The current cursor position will be advanced to the last character put from the save buffer. A <i>count</i> indicates how many copies of the save buffer will be put.                                                                                                                                                                                                                                                                                                                                                                                                |
| 24673 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24674 | <b>[count]P</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24675 |                               | Put a copy of the current contents of the save buffer before the current cursor position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 24676 |                               | The current cursor position will be moved to the last character put from the save buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 24677 |                               | A <i>count</i> indicates how many copies of the save buffer will be put.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 24678 | <b>u</b>                      | Undo the last command that modified the text of the current command line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 24679 | <b>U</b>                      | Undo all changes made to the current command line since first entering command mode on the line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 24680 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24681 | <b>[count]k</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24682 | <b>[count]–</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24683 |                               | Replace the current command line with the previous command line in the shell command history. The cursor will be positioned on the first character of the new command. A count preceding the command will have the same effect as executing the command <i>count</i> times. If a <b>k</b> or <b>–</b> command retreats past the maximum number of commands in effect for this shell (affected by the <i>HISTSIZE</i> environment variable), the terminal will be alerted and the command will have no effect.                                                        |
| 24684 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24685 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24686 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24687 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24688 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24689 | <b>[count]j</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24690 | <b>[count]+</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24691 |                               | Replace the current command line with the next command line in the shell command history. The cursor will be positioned on the first character of the new command. The command history position will be remembered, and any <b>k</b> or <b>–</b> command, or <b>j</b> or <b>+</b> command, will decrement or increment that position and then will fetch the line at the new position. If a <b>j</b> or command advances past the most recent line in the history, the current command line will be restored to the contents before the first <b>k</b> or <b>–</b> . |
| 24692 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24693 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24694 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24695 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24696 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24697 | <b>[number]G</b>              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24698 |                               | Replace the current command line with the contents of the oldest command line stored in the shell command history. With a number <i>number</i> , replace the current command line with the contents of command <i>number</i> in the history.                                                                                                                                                                                                                                                                                                                         |
| 24699 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24700 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24701 | <b>/string&lt;newline&gt;</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24702 |                               | Move backward through the command history, searching for the specified <i>string</i> , beginning with the previous command line. If it is not found, the current command line will be unchanged. If it is found in a previous line, this command will behave equivalently to a set of <b>k</b> commands to reach that line. If <i>string</i> begins with <b>^</b> , the characters after the <b>^</b> will be matched only at the beginning of a line.                                                                                                               |
| 24703 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24704 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24705 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24706 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24707 | <b>?string&lt;newline&gt;</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24708 |                               | Move forward through the command history, searching for the specified string. If it is not found, the current command line will be unchanged. If the string is found in the current command line, the current cursor position will be moved to the beginning of that string. If it is found in the history, this command will behave equivalently to a set of <b>j</b> commands to reach that line. If <i>string</i> begins with <b>^</b> , the characters after the <b>^</b> will be matched only at the beginning of a line.                                       |
| 24709 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24710 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24711 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24712 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24713 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24714 | <b>n</b>                      | Repeat the most recent / or ? command.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 24715 | <b>N</b>                      | Repeat the most recent / or ? command, reversing the direction of the search.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

24716 **EXIT STATUS**

24717 The following exit values are returned:

- 24718           0    The script to be executed consisted solely of zero or more blank lines or comments, or  
 24719                both.  
 24720        1–125    A non-interactive shell detected a syntax, redirection or variable assignment error.  
 24721        127     A specified *command\_file* could not be found by a non-interactive shell.

24722           Otherwise, the shell will return the exit status of the last command it invoked or attempted to  
 24723           invoke (see also the *exit* utility in Section 2.14 on page 67).

24724 **CONSEQUENCES OF ERRORS**

24725 See Section 2.8.1 on page 44.

24726 **APPLICATION USAGE**

24727           Standard input and standard error are the files that determine whether a shell is interactive  
 24728           when **-i** is not specified. For example:

24729           sh &gt; file

24730           and:

24731           sh 2&gt; file

24732           create interactive and non-interactive shells, respectively. Although both accept terminal input,  
 24733           the results of error conditions are different, as described in Section 2.8.1 on page 44; in the second  
 24734           example a redirection error encountered by a special built-in utility will abort the shell.

24735           On systems that support set-user-ID scripts, a historical trapdoor has been to link a script to the  
 24736           name **-i**. When it is called by a sequence such as:

24737           sh -

24738           or by:

24739           #! /bin/sh -

24740           the historical systems have assumed that no option letters follow. Thus, this specification allows  
 24741           the single hyphen to mark the end of the options, in addition to the use of the regular **--**  
 24742           argument, because the older practice is so pervasive.

24743           A portable application must protect its first operand, if it starts with a plus sign, by preceding it  
 24744           with the **--** argument that denotes the end of the options.

24745 **EXAMPLES**

- 24746           1. Execute a shell command from a string:

24747           sh -c "cat myfile"

- 24748           2. Execute a shell script from a file in the current directory:

24749           sh my\_shell\_cmds

24750 **FUTURE DIRECTIONS**

24751           The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 24752           interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 24753           corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 24754           finalised.

24755 **SEE ALSO**

24756           *cd*, *echo*, *pwd*, *test*, *umask*, the **XSH** specification description of *dup()*, *exec*, *exit()*, *fork()*, *pipe()*,  
 24757           *signal()*, *system()*, *ulimit()*, *umask()*, *wait()*.

24758 **CHANGE HISTORY**

24759 First released in Issue 2. |

24760 **Issue 4**24761 Aligned with the ISO/IEC 9945-2: 1993 standard. Description of the shell command language  
24762 and special built-ins moved to Chapter 2 on page 19. |24763 **Issue 5**

24764 FUTURE DIRECTIONS section added. |

24765 Text is added to the DESCRIPTION for the Large File Summit proposal.

24766 **NAME**

24767           sleep — suspend execution for an interval

24768 **SYNOPSIS**

24769           sleep *time*

24770 **DESCRIPTION**

24771           The *sleep* utility will suspend execution for at least the integral number of seconds specified by  
24772           the *time* operand.

24773 **OPTIONS**

24774           None.

24775 **OPERANDS**

24776           The following operands are supported:

24777           *time*       A non-negative decimal integer specifying the number of seconds for which to suspend  
24778                        execution.

24779 **STDIN**

24780           Not used.

24781 **INPUT FILES**

24782           None.

24783 **ENVIRONMENT VARIABLES**

24784           The following environment variables affect the execution of *sleep*:

24785           *LANG*    Provide a default value for the internationalisation variables that are unset or null. If  
24786                        *LANG* is unset or null, the corresponding value from the implementation-dependent  
24787                        default locale will be used. If any of the internationalisation variables contains an  
24788                        invalid setting, the utility will behave as if none of the variables had been defined.

24789           *LC\_ALL*

24790                       If set to a non-empty string value, override the values of all the other  
24791                        internationalisation variables.

24792           *LC\_CTYPE*

24793                       Determine the locale for the interpretation of sequences of bytes of text data as  
24794                        characters (for example, single- as opposed to multi-byte characters in arguments).

24795           *LC\_MESSAGES*

24796                       Determine the locale that should be used to affect the format and contents of diagnostic  
24797                        messages written to standard error.

24798 EX       *NLSPATH*

24799                       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

24800 **ASYNCHRONOUS EVENTS**

24801           If the *sleep* utility receives a SIGALRM signal, one of the following actions will be taken:

- 24802           1. Terminate normally with a zero exit status.
- 24803           2. Effectively ignore the signal.
- 24804           3. Provide the default behaviour for signals described in the ASYNCHRONOUS EVENTS  
24805                        section of Section 1.9 on page 11. This could include terminating with a non-zero exit  
24806                        status.

24807           The *sleep* utility will take the standard action for all other signals.

24808 **STDOUT**

24809 Not used.

24810 **STDERR**

24811 Used only for diagnostic messages.

24812 **OUTPUT FILES**

24813 None.

24814 **EXTENDED DESCRIPTION**

24815 None.

24816 **EXIT STATUS**

24817 The following exit values are returned:

24818       0 The execution was successfully suspended for at least *time* seconds, or a SIGALRM signal  
 24819       was received (see the ASYNCHRONOUS EVENTS section).

24820       &gt;0 An error occurred.

24821 **CONSEQUENCES OF ERRORS**

24822 Default.

24823 **APPLICATION USAGE**

24824 None.

24825 **EXAMPLES**24826 The *sleep* utility can be used to execute a command after a certain amount of time, as in:24827       (*sleep 105; command*) &

24828 or to execute a command every so often, as in:

```
24829 while true
24830 do
24831 command
24832 sleep 37
24833 done
```

24834 **FUTURE DIRECTIONS**

24835 None.

24836 **SEE ALSO**24837 *wait*, the XSH specification description of *alarm()*, *sleep()*.24838 **CHANGE HISTORY**

24839 First released in Issue 2.

24840 **Issue 4**

24841 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 24842 NAME

24843        **sort** — sort, merge or sequence check text files

## 24844 SYNOPSIS

24845 UN        **sort** [-m][-o *output*][-bdfinru][-t *char*][-k *keydef*]...[-z *recsz*]  
24846        [*file*...]

24847 UN        **sort** -c [-bdfinru][-t *char*][-k *keydef*]...[-z *recsz*][*file*...]

24848 OB UN     **sort** [-mu][-o *output*][-bdfir][-t *char*][+pos1[-pos2]]...[-z *recsz*]  
24849        [*file*...]

24850 OB UN     **sort** -c[-u][-bdfinr][-t *char*][+pos1[-pos2]]...[-z *recsz*][*file*]

## 24851 DESCRIPTION

24852        The *sort* utility performs one of the following functions:

- 24853            1. Sorts lines of all the named files together and writes the result to the specified output.
- 24854            2. Merges lines of all the named (presorted) files together and writes the result to the  
24855            specified output.
- 24856            3. Checks that a single input file is correctly presorted.

24857        Comparisons are based on one or more sort keys extracted from each line of input (or the entire  
24858        line if no sort keys are specified), and are performed using the collating sequence of the current  
24859        locale.

## 24860 OPTIONS

24861 OB        The *sort* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**, except  
24862        that the notation *+pos1 -pos2* uses a non-standard prefix and multi-digit option names in the  
24863        obsolescent versions, the **-o output** option is recognised after a *file* operand as an obsolescent  
24864        feature in both versions where the **-c** option is not specified, and the **-k keydef** option should  
24865        follow the **-b**, **-d**, **-f**, **-i**, **-n** and **-r** options.

24866        The following options are supported:

24867        **-c**        Check that the single input file is ordered as specified by the arguments and the  
24868        collating sequence of the current locale. No output is produced; only the exit code is  
24869        affected.

24870        **-m**        Merge only; the input file is assumed to be already sorted.

24871        **-o output**

24872            Specify the name of an output file to be used instead of the standard output. This file  
24873            can be the same as one of the input *files*.

24874        **-u**        Unique: suppress all but one in each set of lines having equal keys. If used with the **-c**  
24875        option, check that there are no lines with duplicate keys, in addition to checking that  
24876        the input file is sorted.

24877 UN        **-z recsz**

24878            The size of the longest line read in the sort phase is recorded so that buffers of the  
24879            correct size can be allocated during the merge phase. If the sort phase is omitted via  
24880            the **-c** or **-m** options, a system-dependent default size will be used. Lines longer than  
24881            the buffer size will cause *sort* to terminate abnormally. Supplying the actual number of  
24882            bytes in the longest line to be merged (or some larger value) will prevent abnormal  
24883            termination.

24884        The following options override the default ordering rules. When ordering options appear  
24885        independent of any key field specifications, the requested field ordering rules are applied

24886 globally to all sort keys. When attached to a specific key (see **-k**), the specified ordering options  
 24887 **OB** override all global ordering options for that key. In the obsolescent forms, if one or more of these  
 24888 options follows a *+pos1* option, it will affect only the key field specified by that preceding option.  
 24889

24890 **-d** Specify that only blank characters and alphanumeric characters, according to the  
 24891 current setting of LC\_CTYPE, are significant in comparisons. The behaviour is  
 24892 undefined for a sort key to which **-i** or **-n** also applies.

24893 **-f** Consider all lower-case characters that have upper-case equivalents, according to the  
 24894 current setting of LC\_CTYPE, to be the upper-case equivalent for the purposes of  
 24895 comparison.

24896 **-i** Ignore all characters that are non-printable, according to the current setting of  
 24897 LC\_CTYPE.

24898 **-n** Restrict the sort key to an initial numeric string, consisting of optional blank characters,  
 24899 optional minus sign, and zero or more digits with an optional radix character and  
 24900 thousands separators (as defined in the current locale), which will be sorted by  
 24901 arithmetic value. An empty digit string is treated as zero. Leading zeros and signs on  
 24902 zeros do not affect ordering.

24903 **-r** Reverse the sense of comparisons.

24904 The treatment of field separators can be altered using the options:

24905 **-b** Ignore leading blank characters when determining the starting and ending positions of  
 24906 a restricted sort key. If the **-b** option is specified before the first **-k** option, it is applied  
 24907 to all **-k** options. Otherwise, the **-b** option can be attached independently to each  
 24908 **-k** *field\_start* or *field\_end* option-argument (see below).

24909 **-t char**  
 24910 Use *char* as the field separator character; *char* is not considered to be part of a field  
 24911 (although it can be included in a sort key). Each occurrence of *char* is significant (for  
 24912 example, *<char><char>* delimits an empty field). If **-t** is not specified, blank characters  
 24913 are used as default field separators; each maximal non-empty sequence of blank  
 24914 characters that follows a non-blank character is a field separator.

24915 Sort keys can be specified using the options:

24916 **-k keydef**  
 24917 The *keydef* argument is a restricted sort key field definition. The format of this  
 24918 definition is:  
 24919 *field\_start*[*type*][,*field\_end*[*type*]]

24920 where *field\_start* and *field\_end* define a key field restricted to a portion of the line (see  
 24921 the EXTENDED DESCRIPTION section), and *type* is a modifier from the list of  
 24922 characters b, d, f, i, n, r. The b modifier behaves like the **-b** option, but applies only to  
 24923 the *field\_start* or *field\_end* to which it is attached. The other modifiers behave like the  
 24924 corresponding options, but apply only to the key field to which they are attached; they  
 24925 have this effect if specified with *field\_start*, *field\_end* or both. If any modifier is attached  
 24926 to a *field\_start* or to a *field\_end*, no option applies to either. Implementations support at  
 24927 least nine occurrences of the **-k** option, which are significant in command line order. If  
 24928 no **-k** option is specified, a default sort key of the entire line is used.

24929 When there are multiple key fields, later keys are compared only after all earlier keys  
 24930 compare equal. Except when the **-u** option is specified, lines that otherwise compare  
 24931 equal are ordered as if none of the options **-d**, **-f**, **-i**, **-n** or **-k** were present (but with **-r**

24932 still in effect, if it was specified) and with all bytes in the lines significant to the  
 24933 comparison. The order in which lines that still compare equal are written is  
 24934 unspecified.

24935 OB **+pos1** Specify the start position of a key field. See the EXTENDED DESCRIPTION section.

24936 OB **-pos2** Specify the end position of a key field. See the EXTENDED DESCRIPTION section.

#### 24937 OPERANDS

24938 The following operand is supported:

24939 **file** A pathname of a file to be sorted, merged or checked. If no *file* operands are specified,  
 24940 or if a *file* operand is "-", the standard input will be used.

#### 24941 STDIN

24942 The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".  
 24943 See the INPUT FILES section.

#### 24944 INPUT FILES

24945 The input files must be text files, except that the *sort* utility will add a newline character to the  
 24946 end of a file ending with an incomplete last line.

#### 24947 ENVIRONMENT VARIABLES

24948 The following environment variables affect the execution of *sort*:

24949 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 24950 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 24951 default locale will be used. If any of the internationalisation variables contains an  
 24952 invalid setting, the utility will behave as if none of the variables had been defined.

24953 **LC\_ALL**

24954 If set to a non-empty string value, override the values of all the other  
 24955 internationalisation variables.

24956 **LC\_COLLATE**

24957 Determine the locale for ordering rules.

24958 **LC\_CTYPE**

24959 Determine the locale for the interpretation of sequences of bytes of text data as  
 24960 characters (for example, single- versus multi-byte characters in arguments and input  
 24961 files) and the behaviour of character classification for the **-b**, **-d**, **-f**, **-i** and **-n** options.

24962 **LC\_MESSAGES**

24963 Determine the locale that should be used to affect the format and contents of diagnostic  
 24964 messages written to standard error.

24965 **LC\_NUMERIC**

24966 Determine the locale for the definition of the radix character and thousands separator  
 24967 for the **-n** option.

24968 EX **NLSPATH**

24969 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

#### 24970 ASYNCHRONOUS EVENTS

24971 Default.

#### 24972 STDOUT

24973 Unless the **-o** or **-c** options are in effect, the standard output contains the sorted input.



24974 **STDERR**

24975       Used for diagnostic messages. A warning message about correcting an incomplete last line of an  
24976       input file may be generated, but need not affect the final exit status.

24977 **OUTPUT FILES**

24978       If the **-o** option is in effect, the sorted input is placed in the file *output*.

24979 **EXTENDED DESCRIPTION**

24980       The notation:

24981        `-k field_start[type][,field_end[type]]`

24982       defines a key field that begins at *field\_start* and ends at *field\_end* inclusive, unless *field\_start* falls  
24983       beyond the end of the line or after *field\_end*, in which case the key field is empty. A missing  
24984       *field\_end* means the last character of the line.

24985       A field comprises a maximal sequence of non-separating characters and, in the absence of option  
24986       **-t**, any preceding field separator.

24987       The *field\_start* portion of the *keydef* option-argument has the form:

24988        `field_number[.first_character]`

24989       Fields and characters within fields are numbered starting with 1. The *field\_number* and  
24990       *first\_character* pieces, interpreted as positive decimal integers, specify the first character to be  
24991       used as part of a sort key. If *first\_character* is omitted, it refers to the first character of the field.

24992       The *field\_end* portion of the *keydef* option-argument has the form:

24993        `field_number[.last_character]`

24994       The *field\_number* is as described above for *field\_start*. The *last\_character* piece, interpreted as a  
24995       non-negative decimal integer, specifies the last character to be used as part of the sort key. If  
24996       *last\_character* evaluates to zero or *last\_character* is omitted, it refers to the last character of the  
24997       field specified by *field\_number*.

24998       If the **-b** option or *b* type modifier is in effect, characters within a field are counted from the first  
24999       non-blank character in the field. (This applies separately to *first\_character* and *last\_character*.)

25000 **OB**       The obsolescent options:

25001        `[+pos1[-pos2]]`

25002       provide functionality equivalent to the **-k** *keydef* option. For comparison, the full formats of  
25003       these options are:

25004        `+field0_number[.first0_character][type]`  
25005        `[-field0_number[.first0_character][type]]`  
25006        `-k field_number[.first_character][type]`  
25007        `[,field_number[.last_character][type]]`

25008       In the obsolescent form, fields (specified by *field0\_number*) and characters within fields (specified  
25009       by *first0\_character*) are numbered from zero instead of one. The optional type modifiers are the  
25010       same in both forms. If *first0\_character* is omitted or *first0\_character* evaluates to zero, it refers to  
25011       the first character of the field. The **-b** option does not apply to *-pos2*.

25012       The fully specified *+pos1 -pos2* form with type modifiers *T* and *U*:

25013        `+w.xT -y.zU`

25014 is equivalent to:

|       |                           |                                                                     |
|-------|---------------------------|---------------------------------------------------------------------|
| 25015 | <i>undefined</i>          | ( <i>z</i> ==0 & <i>U</i> contains <i>b</i> & <i>-t</i> is present) |
| 25016 | <i>-k w+1.x+1T,y.0U</i>   | ( <i>z</i> ==0 otherwise)                                           |
| 25017 | <i>-k w+1.x+1T,y+1.zU</i> | ( <i>z</i> > 0)                                                     |

25018 As with the non-obsolescent forms, implementations support at least nine occurrences of the  
25019 *+pos1* option, which are significant in command line order.

## 25020 EXIT STATUS

25021 The following exit values are returned:

- 25022 0 All input files were output successfully, or *-c* was specified and the input file was correctly
- 25023 sorted.
- 25024 1 Under the *-c* option, the file was not ordered as specified, or if the *-c* and *-u* options were
- 25025 both specified, two input lines were found with equal keys.
- 25026 >1 An error occurred.

## 25027 CONSEQUENCES OF ERRORS

25028 Default.

## 25029 APPLICATION USAGE

25030 The default value for *-t*, blank character, has different properties from, for example,  
25031 *-t "<space>"*. If a line contains:

25032 *<space><space>foo*

25033 the following treatment would occur with default separation as opposed to specifically selecting  
25034 a space character:

| Field | Default                              | <i>-t "&lt;space&gt;"</i> |
|-------|--------------------------------------|---------------------------|
| 1     | <i>&lt;space&gt;&lt;space&gt;foo</i> | <i>empty</i>              |
| 2     | <i>empty</i>                         | <i>empty</i>              |
| 3     | <i>empty</i>                         | <i>foo</i>                |

25040 The leading field separator itself is included in a field when *-t* is not used. For example, this  
25041 command returns an exit status of zero, meaning the input was already sorted:

```
25042 sort -c -k 2 <<eof
25043 y<tab>b
25044 x<space>a
25045 eof
```

25046 (assuming that a tab character precedes the space character in the current collating sequence).  
25047 The field separator is not included in a field when it is explicitly set via *-t*. This is historical  
25048 practice and allows usage such as:

```
25049 sort -t "|" -k 2n <<eof
25050 Atlanta|425022|Georgia
25051 Birmingham|284413|Alabama
25052 Columbia|100385|South Carolina
25053 eof
```

25054 where the second field can be correctly sorted numerically without regard to the non-numeric  
25055 field separator.

25056 The wording in the OPTIONS section clarifies that the **-b**, **-d**, **-f**, **-i**, **-n** and **-r** options have to  
 25057 come before the first sort key specified if they are intended to apply to all specified keys. The  
 25058 way it is described in this document matches historical practice, not historical documentation.  
 25059 In the non-obsolescent versions, the results are unspecified if these options are specified after a  
 25060 **-k** option.

25061 The **-f** option might not work as expected in locales where there is not a one-to-one mapping  
 25062 between an upper- and a lower-case letter.

#### 25063 EXAMPLES

25064 In the following examples, non-obsolescent and obsolescent ways of specifying sort keys are  
 25065 given as an aid to understanding the relationship between the two forms.

- 25066 1. Either of the following commands sorts the contents of **infile** with the second field as the  
 25067 sort key:

25068 `sort -k 2,2 infile`

25069 OB `sort +1 -2 infile`

- 25070 2. Either of the following commands sorts, in reverse order, the contents of **infile1** and  
 25071 **infile2**, placing the output in **outfile** and using the second character of the second field as  
 25072 the sort key (assuming that the first character of the second field is the field separator):

25073 `sort -r -o outfile -k 2.2,2.2 infile1 infile2`

25074 OB `sort -r -o outfile +1.1 -1.2 infile1 infile2`

- 25075 3. Either of the following commands sorts the contents of **infile1** and **infile2** using the  
 25076 second non-blank character of the second field as the sort key:

25077 `sort -k 2.2b,2.2b infile1 infile2`

25078 OB `sort +1.1b -1.2b infile1 infile2`

- 25079 4. Either of the following commands prints the System V password file (user database) sorted  
 25080 by the numeric user ID (the third colon-separated field):

25081 `sort -t : -k 3,3n /etc/passwd`

25082 OB `sort -t : +2 -3n /etc/passwd`

- 25083 5. Either of the following commands prints the lines of the already sorted file **infile**,  
 25084 suppressing all but one occurrence of lines having the same third field:

25085 `sort -um -k 3.1,3.0 infile`

25086 OB `sort -um +2.0 -3.0 infile`

#### 25087 FUTURE DIRECTIONS

25088 None.

#### 25089 SEE ALSO

25090 *comm*, *join*, *uniq*, the XSH specification description of *toupper()*.

#### 25091 CHANGE HISTORY

25092 First released in Issue 2.

#### 25093 Issue 4

25094 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 25095 NAME

25096 spell — find spelling errors (**LEGACY**)

## 25097 SYNOPSIS

25098 EX `spell [-bvx][+local_file][file...]`

## 25099 DESCRIPTION

25100 The *spell* utility collects words from the named files and looks them up in a spelling list. A *word*  
 25101 in this context is a series of characters from the set:

25102 [A-Za-z0-9'&.,;?:]

25103 in the POSIX locale, where the first and last characters are alphanumeric. Words that neither  
 25104 occur among nor are derivable (by applying certain inflections, prefixes and suffixes) from  
 25105 words in the spelling list are written to standard output.

25106 Within the file, certain character sequences are treated specially; if the file contains lines that  
 25107 begin with a period or apostrophe or that contain backslashes in any position, the results are  
 25108 unspecified.

## 25109 OPTIONS

25110 The *spell* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**, except  
 25111 that the *+local\_file* option takes a leading plus sign instead of minus. The following options are  
 25112 supported:

25113 **-v** Write all words not literally in the spelling list. Plausible derivations from the words in  
 25114 the spelling list may be indicated.

25115 **-b** Check British spelling. Besides preferring *centre*, *colour*, *programme*, *speciality*, *travelled*,  
 25116 and so on, this option insists upon *-ise* in words like *standardise*.

25117 **-x** Write every plausible stem with "=" for each word.

25118 *+local\_file*

25119 Remove words found in *local\_file* from the *spell* command output. The argument  
 25120 *local\_file* is the name of a user-provided file that contains a sorted list of words, one per  
 25121 line. With this option, the user can specify a set of words that are correct spellings (in  
 25122 addition to *spell*'s own spelling list) for each job.

## 25123 OPERANDS

25124 The following operands are supported:

25125 *file* A pathname of a text file to check for spelling errors. If no files are named, words are  
 25126 collected from the standard input.

## 25127 STDIN

25128 The standard input is a text file used only if no *file* operands are specified.

## 25129 INPUT FILES

25130 The input files are text files.

## 25131 ENVIRONMENT VARIABLES

25132 The following environment variables may affect the execution of *spell*:

25133 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 25134 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 25135 default locale will be used. If any of the internationalisation variables contains an  
 25136 invalid setting, the utility will behave as if none of the variables had been defined.

25137 *LC\_ALL*  
 25138 If set to a non-empty string value, override the values of all the other  
 25139 internationalisation variables.

25140 *LC\_CTYPE*  
 25141 Determine the locale for the interpretation of sequences of bytes of text data as  
 25142 characters (for example, single- as opposed to multi-byte characters in arguments and  
 25143 input files).

25144 *LC\_MESSAGES*  
 25145 Determine the locale that should be used to affect the format and contents of diagnostic  
 25146 messages written to standard error.

25147 *NLSPATH*  
 25148 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

25149 **ASYNCHRONOUS EVENTS**  
 25150 Default.

25151 **STDOUT**  
 25152 The standard output consists of single misspelled words separated by newlines. If the *-x* option  
 25153 is used, words can be preceded by "=".

25154 **STDERR**  
 25155 Used only for diagnostic messages.

25156 **OUTPUT FILES**  
 25157 None.

25158 **EXTENDED DESCRIPTION**  
 25159 None.

25160 **EXIT STATUS**  
 25161 The following exit values are returned:  
 25162 0 Successful completion.  
 25163 >0 An error occurred.

25164 **CONSEQUENCES OF ERRORS**  
 25165 Default.

25166 **APPLICATION USAGE**  
 25167 None of the internationalisation variables are required to affect the processing of the input  
 25168 source language. In the POSIX locale, the *spell* utility recognises English (American or British  
 25169 dialects, depending on the *-b* option) text.

25170 The unspecified nature of *spell* when presented files with backslashes or leading periods or  
 25171 leading apostrophes results from its complex attempts to deal with files formatted for *troff*, *tbl*  
 25172 and *eqn* processing (none of which are specified by this specification). Constructs such as *.so*,  
 25173 *.nx*, *.TS*, *.EQ*, *.ig*, *\s* and *\f* are frequently dealt with in a manner that is most useful for  
 25174 determining spelling errors. However, such algorithms are historically less than perfect and are  
 25175 very difficult to describe precisely.

25176 This utility is marked **LEGACY** because there is no known technology that can be used to make  
 25177 it recognise general language for user-specified input without providing a complete dictionary  
 25178 along with the input file.

25179 **EXAMPLES**  
 25180 None.

25181 **FUTURE DIRECTIONS**

25182 None.

25183 **SEE ALSO**

25184 None.

25185 **CHANGE HISTORY**

25186 First released in Issue 2.

25187 **Issue 4**

25188 Format reorganised.

25189 Utility Syntax Guidelines support mandated.

25190 Internationalised environment variable support made optional.

25191 Marked TO BE WITHDRAWN.

25192 **Issue 5**

25193 Marked LEGACY.

25194 **NAME**25195 `split` — split files into pieces25196 **SYNOPSIS**25197 `split [-l line_count][-a suffix_length][file[name]]`25198 `split -b n[k|m][-a suffix_length][file[name]]`25199 OB `split [-line_count][-a suffix_length][file[name]]`25200 **DESCRIPTION**

25201 The *split* utility reads an input file and writes one or more output files. The default size of each  
 25202 output file is 1000 lines. The size of the output files can be modified by specification of the **-b** or  
 25203 **-l** options. Each output file is created with a unique suffix. The suffix consists of exactly  
 25204 *suffix\_length* lower-case letters from the POSIX locale. The letters of the suffix are used as if they  
 25205 were a base-26 digit system, with the first suffix to be created consisting of all a characters, the  
 25206 second with a b replacing the last a, and so on, until a name of all z characters is created. By  
 25207 default, the names of the output files are x, followed by a two-character suffix from the character  
 25208 set as described above, starting with aa, ab, ac, and so on, and continuing until the suffix zz, for a  
 25209 maximum of 676 files.

25210 If the number of files required exceeds the maximum allowed by the suffix length provided,  
 25211 such that the last allowable file would be larger than the requested size, the *split* utility will fail  
 25212 after creating the last file with a valid suffix; *split* will not delete the files it created with valid  
 25213 suffixes. If the file limit is not exceeded, the last file created will contain the remainder of the  
 25214 input file, and may be smaller than the requested size.

25215 **OPTIONS**

25216 OB The *split* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines** except  
 25217 that the obsolescent version allows a multi-digit option, **-*line\_count***.

25218 The following options are supported:

25219 **-a *suffix\_length***

25220 Use *suffix\_length* letters to form the suffix portion of the filenames of the split file. If **-a**  
 25221 is not specified, the default suffix length is two. If the sum of the *name* operand and the  
 25222 *suffix\_length* option-argument would create a filename exceeding {NAME\_MAX} bytes,  
 25223 an error will result; *split* will exit with a diagnostic message and no files will be created.

25224 **-b *n*** Split a file into pieces *n* bytes in size.25225 **-b *nk*** Split a file into pieces *n*\*1024 bytes in size.25226 **-b *nm*** Split a file into pieces *n*\*1 048 576 bytes in size.25227 **-l *line\_count***25228 OB **-*line\_count***

25229 Specify the number of lines in each resulting file piece. The *line\_count* argument is an  
 25230 unsigned decimal integer. The default is 1000. If the input does not end with a newline  
 25231 character, the partial line will be included in the last output file.

25232 **OPERANDS**

25233 The following operands are supported:

25234 ***file*** The pathname of the ordinary file to be split. If no input file is given or *file* is "-", the  
 25235 standard input will be used.

25236            *name*      The prefix to be used for each of the files resulting from the split operation. If no *name*  
 25237                            argument is given, x will be used as the prefix of the output files. The combined length  
 25238                            of the basename of *prefix* and *suffix\_length* cannot exceed {NAME\_MAX} bytes; see the  
 25239                            OPTIONS section.

## 25240 STDIN

25241            See the INPUT FILES section.

## 25242 INPUT FILES

25243            Any file can be used as input.

## 25244 ENVIRONMENT VARIABLES

25245            The following environment variables affect the execution of *split*:

25246            *LANG*      Provide a default value for the internationalisation variables that are unset or null. If  
 25247                            *LANG* is unset or null, the corresponding value from the implementation-dependent  
 25248                            default locale will be used. If any of the internationalisation variables contains an  
 25249                            invalid setting, the utility will behave as if none of the variables had been defined.

25250            *LC\_ALL*

25251                            If set to a non-empty string value, override the values of all the other  
 25252                            internationalisation variables.

25253            *LC\_CTYPE*

25254                            Determine the locale for the interpretation of sequences of bytes of text data as  
 25255                            characters (for example, single- as opposed to multi-byte characters in arguments and  
 25256                            input files).

25257            *LC\_MESSAGES*

25258                            Determine the locale that should be used to affect the format and contents of diagnostic  
 25259                            messages written to standard error.

25260 EX        *NLSPATH*

25261                            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 25262 ASYNCHRONOUS EVENTS

25263            Default.

## 25264 STDOUT

25265            Not used.

## 25266 STDERR

25267            Used only for diagnostic messages.

## 25268 OUTPUT FILES

25269            The output files contain portions of the original input file, otherwise unchanged.

## 25270 EXTENDED DESCRIPTION

25271            None.

## 25272 EXIT STATUS

25273            The following exit values are returned:

25274            0      Successful completion.

25275            >0    An error occurred.

## 25276 CONSEQUENCES OF ERRORS

25277            Default.



25278 **APPLICATION USAGE**

25279 None.

25280 **EXAMPLES**

25281 In the following examples foo is a text file that contains 5000 lines.

25282 1. Create five files, xaa, xab, xac, xad and xae:

25283 `split foo` |25284 2. Create five files, but the suffixed portion of the created files consists of three letters, xaaa,  
25285 xaab, xaac, xaad and xaae:25286 `split -a 3 foo` |25287 3. Create three files with four-letter suffixes and a supplied prefix, bar\_aaaa, bar\_aaab and  
25288 bar\_aaac:25289 `split -a 4 -l 2000 foo bar_` |25290 4. Create as many files as are necessary to contain at most 20\*1024 bytes, each with the  
25291 default prefix of **x** and a five-letter suffix:25292 `split -a 5 -b 20k foo` |25293 **FUTURE DIRECTIONS**

25294 None.

25295 **SEE ALSO**25296 *csplit*.25297 **CHANGE HISTORY**

25298 First released in Issue 2. |

25299 **Issue 4**

25300 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 25301 NAME

25302 strings — find printable strings in files

## 25303 SYNOPSIS

25304 strings [-a][-t *format*][-n *number*][*file...*]

25305 OB strings [-][-t *format*][-*number*][*file...*]

## 25306 DESCRIPTION

25307 The *strings* utility looks for printable strings in regular files and writes those strings to standard  
 25308 output. A printable string is any sequence of four (by default) or more printable characters  
 25309 terminated by a newline or NUL character. Additional implementation-dependent strings may  
 25310 be written. (See *localedef*.)

## 25311 OPTIONS

25312 OB The *strings* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**  
 25313 except that the obsolescent version uses "-" in a non-standard way and allows a multi-digit  
 25314 option, *-number*.

25315 The following options are supported:

25316 **-a**

25317 OB - Scan files in their entirety. If **-a** is not specified, it is implementation-dependent what  
 25318 portion of each file is scanned for strings.

25319 **-n *number***

25320 OB **-*number***

25321 Specify the minimum string length, where the *number* argument is a positive decimal  
 25322 integer. The default is 4.

25323 **-t *format***

25324 Write each string preceded by its byte offset from the start of the file. The format is  
 25325 dependent on the single character used as the *format* option-argument:

25326 **d** The offset will be written in decimal.

25327 **o** The offset will be written in octal.

25328 **x** The offset will be written in hexadecimal.

## 25329 OPERANDS

25330 The following operand is supported:

25331 ***file*** A pathname of a regular file to be used as input. If no *file* operand is specified, the  
 25332 *strings* utility will read from the standard input.

## 25333 STDIN

25334 See the INPUT FILES section.

## 25335 INPUT FILES

25336 The input files named by the utility arguments or the standard input must be regular files of any  
 25337 format.

## 25338 ENVIRONMENT VARIABLES

25339 The following environment variables affect the execution of *strings*:

25340 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 25341 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 25342 default locale will be used. If any of the internationalisation variables contains an  
 25343 invalid setting, the utility will behave as if none of the variables had been defined.

25344 **LC\_ALL**  
 25345 If set to a non-empty string value, override the values of all the other  
 25346 internationalisation variables.

25347 **LC\_CTYPE**  
 25348 Determine the locale for the interpretation of sequences of bytes of text data as  
 25349 characters (for example, single- as opposed to multi-byte characters in arguments and  
 25350 input files) and to identify printable strings.

25351 **LC\_MESSAGES**  
 25352 Determine the locale that should be used to affect the format and contents of diagnostic  
 25353 messages written to standard error.

25354 EX **NLSPATH**  
 25355 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

25356 **ASYNCHRONOUS EVENTS**  
 25357 Default.

25358 **STDOUT**  
 25359 Strings found are written to the standard output, one per line.  
 25360 When the **-t** option is not specified, the format of the output is:  
 25361 `"%s", <string>`  
 25362 With the **-t o** option, the format of the output is:  
 25363 `"%o %s", <byte offset>, <string>`  
 25364 With the **-t x** option, the format of the output is:  
 25365 `"%x %s", <byte offset>, <string>`  
 25366 With the **-t d** option, the format of the output is:  
 25367 `"%d %s", <byte offset>, <string>`

25368 **STDERR**  
 25369 Used only for diagnostic messages.

25370 **OUTPUT FILES**  
 25371 None.

25372 **EXTENDED DESCRIPTION**  
 25373 None.

25374 **EXIT STATUS**  
 25375 The following exit values are returned:  
 25376 0 Successful completion.  
 25377 >0 An error occurred.

25378 **CONSEQUENCES OF ERRORS**  
 25379 Default.

25380 **APPLICATION USAGE**  
 25381 By default the data area (as opposed to the text, “bss” or header areas) of a binary executable file  
 25382 is scanned. Implementations will document which areas are scanned.  
 25383 Some historical implementations do not require NUL or newline character terminators for  
 25384 strings to permit those languages that do not use NUL as a string terminator to have their strings  
 25385 written.

**25386 EXAMPLES**

25387           None.

**25388 FUTURE DIRECTIONS**

25389           None.

**25390 SEE ALSO**

25391           *nm*.

**25392 CHANGE HISTORY**

25393           First released in Issue 4.

25394 **NAME**25395 strip — remove unnecessary information from executable files (**DEVELOPMENT**)25396 **SYNOPSIS**25397 strip *file...*25398 **DESCRIPTION**

25399 The *strip* utility removes from executable files named by the *file* operands any information the  
 25400 implementor deems unnecessary for execution of those files. The nature of that information is  
 25401 unspecified. The effect of *strip* is the same as the use of the **-s** option to *cc*, *c89* or *fort77*.

25402 **OPTIONS**

25403 None.

25404 **OPERANDS**

25405 The following operand is supported:

25406 *file* A pathname referring to an executable file.25407 **STDIN**

25408 Not used.

25409 **INPUT FILES**

25410 The input files must be in the form of executable files successfully produced by any compiler  
 25411 defined by this specification.

25412 **ENVIRONMENT VARIABLES**25413 The following environment variables affect the execution of *strip*:

25414 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 25415 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 25416 default locale will be used. If any of the internationalisation variables contains an  
 25417 invalid setting, the utility will behave as if none of the variables had been defined.

25418 **LC\_ALL**

25419 If set to a non-empty string value, override the values of all the other  
 25420 internationalisation variables.

25421 **LC\_CTYPE**

25422 Determine the locale for the interpretation of sequences of bytes of text data as  
 25423 characters (for example, single- as opposed to multi-byte characters in arguments).

25424 **LC\_MESSAGES**

25425 Determine the locale that should be used to affect the format and contents of diagnostic  
 25426 messages written to standard error.

25427 EX **NLSPATH**25428 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.25429 **ASYNCHRONOUS EVENTS**

25430 Default.

25431 **STDOUT**

25432 Not used.

25433 **STDERR**

25434 Used only for diagnostic messages.

25435 **OUTPUT FILES**25436 The *strip* utility will produce executable files of unspecified format.

25437 **EXTENDED DESCRIPTION**

25438           None.

25439 **EXIT STATUS**

25440           The following exit values are returned:

25441           0   Successful completion.

25442           &gt;0  An error occurred.

25443 **CONSEQUENCES OF ERRORS**

25444           Default.

25445 **APPLICATION USAGE**

25446           None.

25447 **EXAMPLES**

25448           None.

25449 **FUTURE DIRECTIONS**

25450           None.

25451 **SEE ALSO**25452           *ar, cc, c89, fort77.*25453 **CHANGE HISTORY**

25454           First released in Issue 2. |

25455 **Issue 4**

25456           Aligned with the ISO/IEC 9945-2: 1993 standard.

25457 **NAME**

25458           stty — set the options for a terminal

25459 **SYNOPSIS**

25460           stty [ -a | -g ]

25461           stty *operands*25462 **DESCRIPTION**

25463       The *stty* utility sets or reports on terminal I/O characteristics for the device that is its standard  
 25464       input. Without options or operands specified, it reports the settings of certain characteristics,  
 25465       usually those that differ from implementation-dependent defaults. Otherwise, it modifies the  
 25466       terminal state according to the specified operands. Detailed information about the modes listed  
 25467       in the first five groups below are described in the **XBD** specification, **Chapter 9, General**  
 25468       **Terminal Interface**. Operands in the Combination Modes group (see **Combination Modes** on  
 25469       page 693) are implemented using operands in the previous groups. Some combinations of  
 25470       operands are mutually exclusive on some terminal types; the results of using such combinations  
 25471       are unspecified.

25472       Typical implementations of this utility require a communications line configured to use a **XSH**  
 25473       specification **termios** interface. On systems where none of these lines are available, and on lines  
 25474       not currently configured to support the **XSH** specification **termios** interface, some of the  
 25475       operands need not affect terminal characteristics.

25476 **OPTIONS**25477       The *stty* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

25478       The following options are supported:

25479       **-a**       Write to standard output all the current settings for the terminal.

25480       **-g**       Write to standard output all the current settings in an unspecified form that can be used  
 25481       as arguments to another invocation of the *stty* utility on the same system. The form  
 25482       used will not contain any characters that would require quoting to avoid word  
 25483       expansion by the shell; see Section 2.6 on page 31.

25484 **OPERANDS**

25485       The following operands are supported to set the terminal characteristics:

25486       **Control Modes**25487       **parenb** (**-parenb**)

25488       Enable (disable) parity generation and detection. This has the effect of setting (not  
 25489       setting) PARENB in the **termios c\_cflag** field, as defined in the **XBD** specification,  
 25490       **Chapter 9, General Terminal Interface**.

25491       **parodd** (**-parodd**)

25492       Select odd (even) parity. This has the effect of setting (not setting) PARODD in the  
 25493       **termios c\_cflag** field, as defined in the **XBD** specification, **Chapter 9, General Terminal**  
 25494       **Interface**.

25495       **cs5 cs6 cs7 cs8**

25496       Select character size, if possible. This has the effect of setting CS5, CS6, CS7 and CS8,  
 25497       respectively, in the **termios c\_cflag** field, as defined in the **XBD** specification,  
 25498       **Chapter 9, General Terminal Interface**.

25499        *number*    Set terminal baud rate to the number given, if possible. If the baud rate is set to zero,  
 25500        the modem control lines will no longer be asserted. This has the effect of setting the  
 25501        input and output **termios** baud rate values as defined in the **XBD** specification,  
 25502        **Chapter 9, General Terminal Interface**.

25503        **ispeed** *number*  
 25504        Set terminal input baud rate to the number given, if possible. If the input baud rate is  
 25505        set to zero, the input baud rate will be specified by the value of the output baud rate.  
 25506        This has the effect of setting the input **termios** baud rate values as defined in the **XBD**  
 25507        specification, **Chapter 9, General Terminal Interface**.

25508        **ospeed** *number*  
 25509        Set terminal output baud rate to the number given, if possible. If the output baud rate  
 25510        is set to zero, the modem control lines will no longer be asserted. This has the effect of  
 25511        setting the output **termios** baud rate values as defined in the **XBD** specification,  
 25512        **Chapter 9, General Terminal Interface**.

25513        **hupcl** (**-hupcl**)  
 25514        Stop asserting modem control lines (do not stop asserting modem control lines) on last  
 25515        close. This has the effect of setting (not setting) HUPCL in the **termios c\_cflag** field, as  
 25516        defined in the **XBD** specification, **Chapter 9, General Terminal Interface**.

25517        **hup** (**-hup**)  
 25518        Same as **hupcl** (**-hupcl**).

25519        **cstopb** (**-cstopb**)  
 25520        Use two (one) stop bits per character. This has the effect of setting (not setting)  
 25521        CSTOPB in the **termios c\_cflag** field, as defined in the **XBD** specification, **Chapter 9**,  
 25522        **General Terminal Interface**.

25523        **cread** (**-cread**)  
 25524        Enable (disable) the receiver. This has the effect of setting (not setting) CREAD in the  
 25525        **termios c\_cflag** field, as defined in the **XBD** specification, **Chapter 9, General Terminal**  
 25526        **Interface**.

25527        **clocal** (**-clocal**)  
 25528        Assume a line without (with) modem control. This has the effect of setting (not setting)  
 25529        CLOCAL in the **termios c\_cflag** field, as defined in the **XBD** specification, **Chapter 9**,  
 25530        **General Terminal Interface**.

25531        It is unspecified whether *stty* will report an error if an attempt to set a Control Mode fails.

25532        **Input Modes**

25533        **ignbrk** (**-ignbrk**)  
 25534        Ignore (do not ignore) break on input. This has the effect of setting (not setting)  
 25535        IGNBRK in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9**,  
 25536        **General Terminal Interface**.

25537        **brkint** (**-brkint**)  
 25538        Signal (do not signal) INTR on break. This has the effect of setting (not setting)  
 25539        BRKINT in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9**,  
 25540        **General Terminal Interface**.

25541        **ignpar** (**-ignpar**)  
 25542        Ignore (do not ignore) bytes with parity errors. This has the effect of setting (not  
 25543        setting) IGNPAR in the **termios c\_iflag** field, as defined in the **XBD** specification,  
 25544        **Chapter 9, General Terminal Interface**.



25545 **parmrk** (**-parmrk**)  
 25546 Mark (do not mark) parity errors. This has the effect of setting (not setting) PARMRK  
 25547 in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9, General**  
 25548 **Terminal Interface**.

25549 **inpck** (**-inpck**)  
 25550 Enable (disable) input parity checking. This has the effect of setting (not setting)  
 25551 INPCK in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9,**  
 25552 **General Terminal Interface**.

25553 **istrip** (**-istrip**)  
 25554 Strip (do not strip) input characters to seven bits. This has the effect of setting (not  
 25555 setting) ISTRIP in the **termios c\_iflag** field, as defined in the **XBD** specification,  
 25556 **Chapter 9, General Terminal Interface**.

25557 **inlcr** (**-inlcr**)  
 25558 Map (do not map) NL to CR on input. This has the effect of setting (not setting) INLCR  
 25559 in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9, General**  
 25560 **Terminal Interface**.

25561 **igncr** (**-igncr**)  
 25562 Ignore (do not ignore) CR on input. This has the effect of setting (not setting) IGNCR  
 25563 in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9, General**  
 25564 **Terminal Interface**.

25565 **icrnl** (**-icrnl**)  
 25566 Map (do not map) CR to NL on input. This has the effect of setting (not setting) ICRNL  
 25567 in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9, General**  
 25568 **Terminal Interface**.

25569 **iucLC** (**-iucLC**)  
 25570 EX Map (do not map) upper-case alphabets to lower-case on input. This has the effect of  
 25571 setting (not setting) IUCLC in the **termios c\_iflag** field, as defined in the **XBD**  
 25572 specification, **Chapter 9, General Terminal Interface**. (LEGACY)

25573 **ixon** (**-ixon**)  
 25574 Enable (disable) START/STOP output control. Output from the system is stopped  
 25575 when the system receives STOP and started when the system receives START. This has  
 25576 the effect of setting (not setting) IXON in the **termios c\_iflag** field, as defined in the  
 25577 **XBD** specification, **Chapter 9, General Terminal Interface**.

25578 **ixany** (**-ixany**)  
 25579 EX Allow any character to restart output. This has the effect of setting (not setting) IXANY  
 25580 in the **termios c\_iflag** field, as defined in the **XBD** specification, **Chapter 9, General**  
 25581 **Terminal Interface**.

25582 **ixoff** (**-ixoff**)  
 25583 Request that the system send (not send) STOP characters when the input queue is  
 25584 nearly full and START characters to resume data transmission. This has the effect of  
 25585 setting (not setting) IXOFF in the **termios c\_iflag** field, as defined in the **XBD**  
 25586 specification, **Chapter 9, General Terminal Interface**.

|          |                                                                                                                 |  |
|----------|-----------------------------------------------------------------------------------------------------------------|--|
| 25587    | <b>Output Modes</b>                                                                                             |  |
| 25588    | <b>opost (-opost)</b>                                                                                           |  |
| 25589    | Post-process output (do not post-process output; ignore all other output modes). This                           |  |
| 25590    | has the effect of setting (not setting) OPOST in the <b>termios</b> <i>c_oflag</i> field, as defined in         |  |
| 25591    | the <b>XBD</b> specification, <b>Chapter 9, General Terminal Interface</b> .                                    |  |
| 25592 EX | <b>olcuc (-olcuc)</b>                                                                                           |  |
| 25593    | Map (do not map) lower-case alphabets to upper-case on output. This has the effect                              |  |
| 25594    | of setting (not setting) OLCUC in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b>         |  |
| 25595    | specification, <b>Chapter 9, General Terminal Interface</b> . (LEGACY)                                          |  |
| 25596    | <b>ocrnl (-ocrnl)</b>                                                                                           |  |
| 25597    | Map (do not map) CR to NL on output This has the effect of setting (not setting)                                |  |
| 25598    | OCRNL in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b> specification, <b>Chapter 9,</b> |  |
| 25599    | <b>General Terminal Interface</b> .                                                                             |  |
| 25600    | <b>onocr (-onocr)</b>                                                                                           |  |
| 25601    | Do not (do) output CR at column zero. This has the effect of setting (not setting)                              |  |
| 25602    | ONOCR in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b> specification, <b>Chapter 9,</b> |  |
| 25603    | <b>General Terminal Interface</b> .                                                                             |  |
| 25604    | <b>onlret (-onlret)</b>                                                                                         |  |
| 25605    | The terminal newline key performs (does not perform) the CR function. This has the                              |  |
| 25606    | effect of setting (not setting) ONLRET in the <b>termios</b> <i>c_oflag</i> field, as defined in the            |  |
| 25607    | <b>XBD</b> specification, <b>Chapter 9, General Terminal Interface</b> .                                        |  |
| 25608    | <b>ofill (-ofill)</b>                                                                                           |  |
| 25609    | Use fill characters (use timing) for delays. This has the effect of setting (not setting)                       |  |
| 25610    | OFILL in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b> specification, <b>Chapter 9,</b> |  |
| 25611    | <b>General Terminal Interface</b> .                                                                             |  |
| 25612    | <b>ofdel (-ofdel)</b>                                                                                           |  |
| 25613    | Fill characters are DELs (NULs). This has the effect of setting (not setting) OFDEL in                          |  |
| 25614    | the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b> specification, <b>Chapter 9, General</b>  |  |
| 25615    | <b>Terminal Interface</b> .                                                                                     |  |
| 25616    | <b>cr0 cr1 cr2 cr3</b>                                                                                          |  |
| 25617    | Select the style of delay for CRs. This has the effect of setting (not setting) CRDLY to                        |  |
| 25618    | CR1, CR2, CR3 or CR4, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b>    |  |
| 25619    | specification, <b>Chapter 9, General Terminal Interface</b> .                                                   |  |
| 25620    | <b>nl0 nl1</b> Select the style of delay for NL. This has the effect of setting (not setting) NLDLY to          |  |
| 25621    | NL0 or NL1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b>              |  |
| 25622    | specification, <b>Chapter 9, General Terminal Interface</b> .                                                   |  |
| 25623    | <b>tab0 tab1 tab2 tab3</b>                                                                                      |  |
| 25624    | Select the style of delay for horizontal tabs. This has the effect of setting (not setting)                     |  |
| 25625    | TABDLY to TAB0, TAB1, TAB2 or TAB3, respectively, in the <b>termios</b> <i>c_oflag</i> field, as                |  |
| 25626    | defined in the <b>XBD</b> specification, <b>Chapter 9, General Terminal Interface</b> . Note that               |  |
| 25627    | TAB3 has the effect of expanding tabs to spaces.                                                                |  |
| 25628    | <b>bs0 bs1</b> Select the style of delay for backspaces. This has the effect of setting (not setting)           |  |
| 25629    | BSDLY to BS0 or BS1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b>     |  |
| 25630    | specification, <b>Chapter 9, General Terminal Interface</b> .                                                   |  |
| 25631    | <b>ff0 ff1</b> Select the style of delay for form-feeds. This has the effect of setting (not setting)           |  |
| 25632    | FFDLY to FF0 or FF1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the <b>XBD</b>     |  |

|          |                                                                                                                         |  |
|----------|-------------------------------------------------------------------------------------------------------------------------|--|
| 25633    | specification, <b>Chapter 9, General Terminal Interface.</b>                                                            |  |
| 25634    | <b>vt0 vt1</b> Select the style of delay for vertical-tabs. This has the effect of setting (not setting)                |  |
| 25635    | VTDLY to VT0 or VT1, respectively, in the <b>termios c_oflag</b> field, as defined in the <b>XBD</b>                    |  |
| 25636    | specification, <b>Chapter 9, General Terminal Interface.</b>                                                            |  |
| 25637    | <b>Local Modes</b>                                                                                                      |  |
| 25638    | <b>isig (-isig)</b>                                                                                                     |  |
| 25639    | Enable (disable) the checking of characters against the special control characters INTR,                                |  |
| 25640    | QUIT, and SUSP. This has the effect of setting (not setting) ISIG in the <b>termios c_lflag</b>                         |  |
| 25641    | field, as defined in the <b>XBD</b> specification, <b>Chapter 9, General Terminal Interface.</b>                        |  |
| 25642    | <b>icanon (-icanon)</b>                                                                                                 |  |
| 25643    | Enable (disable) canonical input (ERASE and KILL processing). This has the effect of                                    |  |
| 25644    | setting (not setting) ICANON in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b>                          |  |
| 25645    | specification, <b>Chapter 9, General Terminal Interface.</b>                                                            |  |
| 25646    | <b>xcase (-xcase)</b>                                                                                                   |  |
| 25647 EX | Set canonical (unprocessed) upper- or lower-case presentation. This has the effect of                                   |  |
| 25648    | setting (not setting) XCASE in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b>                           |  |
| 25649    | specification, <b>Chapter 9, General Terminal Interface. (LEGACY)</b>                                                   |  |
| 25650    | <b>ixexten (-ixexten)</b>                                                                                               |  |
| 25651    | Enable (disable) any implementation-dependent special control characters not                                            |  |
| 25652    | currently controlled by <b>icanon</b> , <b>isig</b> , <b>ixon</b> or <b>ixoff</b> . This has the effect of setting (not |  |
| 25653    | setting) IEXTEN in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification,                        |  |
| 25654    | <b>Chapter 9, General Terminal Interface.</b>                                                                           |  |
| 25655    | <b>echo (-echo)</b>                                                                                                     |  |
| 25656    | Echo back (do not echo back) every character typed. This has the effect of setting (not                                 |  |
| 25657    | setting) ECHO in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification,                          |  |
| 25658    | <b>Chapter 9, General Terminal Interface.</b>                                                                           |  |
| 25659    | <b>echoe (-echoe)</b>                                                                                                   |  |
| 25660    | The ERASE character will (will not) visually erase the last character in the current line                               |  |
| 25661    | from the display, if possible. This has the effect of setting (not setting) ECHOE in the                                |  |
| 25662    | <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification, <b>Chapter 9, General Terminal</b>            |  |
| 25663    | <b>Interface.</b>                                                                                                       |  |
| 25664    | <b>echok (-echok)</b>                                                                                                   |  |
| 25665    | Echo (do not echo) NL after KILL character. This has the effect of setting (not setting)                                |  |
| 25666    | ECHOK in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification, <b>Chapter 9,</b>                |  |
| 25667    | <b>General Terminal Interface.</b>                                                                                      |  |
| 25668    | <b>echonl (-echonl)</b>                                                                                                 |  |
| 25669    | Echo (do not echo) NL, even if <b>echo</b> is disabled. This has the effect of setting (not                             |  |
| 25670    | setting) ECHONL in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification,                        |  |
| 25671    | <b>Chapter 9, General Terminal Interface.</b>                                                                           |  |
| 25672    | <b>noflsh (-noflsh)</b>                                                                                                 |  |
| 25673    | Disable (enable) flush after INTR, QUIT, SUSP. This has the effect of setting (not                                      |  |
| 25674    | setting) NOFLSH in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification,                        |  |
| 25675    | <b>Chapter 9, General Terminal Interface.</b>                                                                           |  |
| 25676    | <b>tostop (-tostop)</b>                                                                                                 |  |
| 25677    | Send SIGTTOU for background output. This has the effect of setting (not setting)                                        |  |
| 25678    | TOSTOP in the <b>termios c_lflag</b> field, as defined in the <b>XBD</b> specification, <b>Chapter 9,</b>               |  |

## General Terminal Interface.

**Note:** Setting TOSTOP has no effect on systems not supporting the job control option, but all XSI-conformant systems do support this option.

## Special Control Character Assignments

*<control>-character string*

Set *<control>-character* to *string*. If *<control>-character* is one of the character sequences in the first column of the following table, the corresponding **XBD** specification, **Chapter 9, General Terminal Interface** control character from the second column will be recognised. This has the effect of setting the corresponding element of the **termios** **c\_cc** array (see the **XSH** specification **<termios.h>**).

| Control Character | c_cc Subscript | Description     |
|-------------------|----------------|-----------------|
| eof               | VEOF           | EOF character   |
| eol               | VEOL           | EOL character   |
| erase             | VERASE         | ERASE character |
| intr              | VINTR          | INTR character  |
| kill              | VKILL          | KILL character  |
| quit              | VQUIT          | QUIT character  |
| susp              | VSUSP          | SUSP character  |
| start             | VSTART         | START character |
| stop              | VSTOP          | STOP character  |

**Table 3-15** Control Character Names in *stty*

If *string* is a single character, the control character will be set to that character. If *string* is the two-character sequence "^-" or the string undef, the control character will be set to **{\_POSIX\_VDISABLE}**, if it is in effect for the device; if **{\_POSIX\_VDISABLE}** is not in effect for the device, it will be treated as an error. In the POSIX locale, if *string* is a two-character sequence beginning with circumflex (^), and the second character is one of those listed in the ^c column of the following table, the control character will be set to the corresponding character value in the Value column of the table.

| ^c   | Value | ^c   | Value | ^c   | Value |
|------|-------|------|-------|------|-------|
| a, A | <SOH> | l, L | <FF>  | w, W | <ETB> |
| b, B | <STX> | m, M | <CR>  | x, X | <CAN> |
| c, C | <ETX> | n, N | <SO>  | y, Y | <EM>  |
| d, D | <EOT> | o, O | <SI>  | z, Z | <SUB> |
| e, E | <ENQ> | p, P | <DLE> | [    | <ESC> |
| f, F | <ACK> | q, Q | <DC1> | \    | <FS>  |
| g, G | <BEL> | r, R | <DC2> | ]    | <GS>  |
| h, H | <BS>  | s, S | <DC3> | ^    | <RS>  |
| i, I | <HT>  | t, T | <DC4> | _    | <US>  |
| j, J | <LF>  | u, U | <NAK> | ?    | <DEL> |
| k, K | <VT>  | v, V | <SYN> |      |       |

**Table 3-16** Circumflex Control Characters in *stty*

|          |                                                |               |                                                                                                           |
|----------|------------------------------------------------|---------------|-----------------------------------------------------------------------------------------------------------|
| 25722    | <b>min</b>                                     | <i>number</i> |                                                                                                           |
| 25723    | <b>time</b>                                    | <i>number</i> |                                                                                                           |
| 25724    |                                                |               | Set the value of <b>min</b> or <b>time</b> to <i>number</i> . MIN and TIME are used in non-canonical mode |
| 25725    |                                                |               | input processing ( <b>-icanon</b> ).                                                                      |
| 25726    | <b>Combination Modes</b>                       |               |                                                                                                           |
| 25727    | <i>saved settings</i>                          |               |                                                                                                           |
| 25728    |                                                |               | Set the current terminal characteristics to the saved settings produced by the <b>-g</b> option.          |
| 25729    | <b>evenp</b> or <b>parity</b>                  |               |                                                                                                           |
| 25730    |                                                |               | Enable <b>parenb</b> and <b>cs7</b> ; disable <b>parodd</b> .                                             |
| 25731    | <b>oddp</b>                                    |               | Enable <b>parenb</b> , <b>cs7</b> and <b>parodd</b> .                                                     |
| 25732    | <b>-parity</b> , <b>-evenp</b> or <b>-oddp</b> |               |                                                                                                           |
| 25733    |                                                |               | Disable <b>parenb</b> , and set <b>cs8</b> .                                                              |
| 25734 EX | <b>raw</b> ( <b>-raw</b> or <b>cooked</b> )    |               |                                                                                                           |
| 25735    |                                                |               | Enable (disable) raw input and output. Raw mode is equivalent to setting:                                 |
| 25736    |                                                |               | <code>stty cs8 erase ^- kill ^- intr ^- \</code>                                                          |
| 25737    |                                                |               | <code>quit ^- eof ^- eol ^- -opost -inpck</code>                                                          |
| 25738    | <b>nl</b> ( <b>-nl</b> )                       |               | Enable (disable) <b>icrnl</b> . In addition, <b>-nl</b> unsets <b>inlcr</b> and <b>igncr</b> .            |
| 25739 EX | <b>lcase</b> ( <b>-lcase</b> )                 |               |                                                                                                           |
| 25740    |                                                |               | Set (unset) <b>xcase</b> , <b>iucLC</b> and <b>olcuc</b> . ( <b>LEGACY</b> )                              |
| 25741 EX | <b>LCASE</b> ( <b>-LCASE</b> )                 |               |                                                                                                           |
| 25742    |                                                |               | Equivalent to <b>lcase</b> ( <b>-lcase</b> ). ( <b>LEGACY</b> )                                           |
| 25743 EX | <b>tabs</b> ( <b>-tabs</b> or <b>tab3</b> )    |               |                                                                                                           |
| 25744    |                                                |               | Preserve tabs (expand to spaces) when printing.                                                           |
| 25745    | <b>ek</b>                                      |               | Reset ERASE and KILL characters back to system defaults.                                                  |
| 25746    | <b>sane</b>                                    |               | Reset all modes to some reasonable, unspecified, values.                                                  |
| 25747    | <b>STDIN</b>                                   |               |                                                                                                           |
| 25748    |                                                |               | Although no input is read from standard input, standard input is used to get the current                  |
| 25749    |                                                |               | terminal I/O characteristics and to set new terminal I/O characteristics.                                 |
| 25750    | <b>INPUT FILES</b>                             |               |                                                                                                           |
| 25751    |                                                |               | None.                                                                                                     |
| 25752    | <b>ENVIRONMENT VARIABLES</b>                   |               |                                                                                                           |
| 25753    |                                                |               | The following environment variables affect the execution of <i>stty</i> :                                 |
| 25754    | <b>LANG</b>                                    |               | Provide a default value for the internationalisation variables that are unset or null. If                 |
| 25755    |                                                |               | <b>LANG</b> is unset or null, the corresponding value from the implementation-dependent                   |
| 25756    |                                                |               | default locale will be used. If any of the internationalisation variables contains an                     |
| 25757    |                                                |               | invalid setting, the utility will behave as if none of the variables had been defined.                    |
| 25758    | <b>LC_ALL</b>                                  |               |                                                                                                           |
| 25759    |                                                |               | If set to a non-empty string value, override the values of all the other                                  |
| 25760    |                                                |               | internationalisation variables.                                                                           |
| 25761    | <b>LC_CTYPE</b>                                |               |                                                                                                           |
| 25762    |                                                |               | This variable will determine the locale for the interpretation of sequences of bytes of                   |
| 25763    |                                                |               | text data as characters (for example, single- versus multi-byte characters in arguments)                  |

25764 and which characters are in the class **print**.

25765 **LC\_MESSAGES**

25766 Determine the locale that should be used to affect the format and contents of diagnostic

25767 messages written to standard error.

25768 EX **NLSPATH**

25769 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

25770 **ASYNCHRONOUS EVENTS**

25771 Default.

25772 **STDOUT**

25773 If operands are specified, no output is produced.

25774 If the **-g** option is specified, **stty** will write to standard output the current settings in a form that

25775 can be used as arguments to another instance of **stty** on the same system.

25776 If the **-a** option is specified, all of the information as described in the OPERANDS section will be

25777 written to standard output. Unless otherwise specified, this information is written as space-

25778 separated tokens in an unspecified format, on one or more lines, with an unspecified number of

25779 tokens per line. Additional information may be written.

25780 If no options or operands are specified, an unspecified subset of the information written for the

25781 **-a** option is written.

25782 If speed information is written as part of the default output, or if the **-a** option is specified and if

25783 the terminal input speed and output speed are the same, the speed information will be written

25784 as follows:

25785 "speed %d baud;", <speed>

25786 Otherwise, speeds will be written as:

25787 "ispeed %d baud; ospeed %d baud;", <ispeed>, <ospeed>

25788 In locales other than the POSIX locale, the word **baud** may be changed to something more

25789 appropriate in those locales.

25790 If control characters are written as part of the default output, or if the **-a** option is specified,

25791 control characters will be written as:

25792 "%s = %s;", <<control>-character name>, <value>

25793 where *value* is either the character, or some visual representation of the character if it is non-

25794 printable, or the string <undef> if the character is disabled.

25795 **STDERR**

25796 Used only for diagnostic messages.

25797 **OUTPUT FILES**

25798 None.

25799 **EXTENDED DESCRIPTION**

25800 None.

25801 **EXIT STATUS**

25802 The following exit values are returned:

25803 0 The terminal options were read or set successfully.

25804 >0 An error occurred.

25805 **CONSEQUENCES OF ERRORS**

25806 Default.

25807 **APPLICATION USAGE**25808 The **-g** flag is designed to facilitate the saving and restoring of terminal state from the shell level.

25809 For example, a program may:

```

25810 saveterm="$(stty -g)" # save terminal state
25811 stty (new settings) # set new state
25812 ... # ...
25813 stty $saveterm # restore terminal state

```

25814 Since the format is unspecified, the saved value is not portable across systems.

25815 Since the **-a** format is so loosely specified, scripts that save and restore terminal settings should  
 25816 use the **-g** option.

25817 **EXAMPLES**

25818 None.

25819 **FUTURE DIRECTIONS**

25820 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 25821 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 25822 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 25823 finalised.

25824 **SEE ALSO**25825 The **XBD** specification, **Chapter 9, General Terminal Interface**.25826 **CHANGE HISTORY**

25827 First released in Issue 2.

25828 **Issue 4**

25829 Aligned with the ISO/IEC 9945-2: 1993 standard.

25830 **Issue 5**25831 The description of **tabs** is clarified.

25832 FUTURE DIRECTIONS section added.

25833 **NAME**

25834           sum — print checksum and block count of a file (**LEGACY**)

25835 **SYNOPSIS**

25836 EX       sum [-r][*file...*]

25837 **DESCRIPTION**

25838           The *sum* utility calculates and writes a checksum for the named file to standard output and also  
 25839           writes the space used by the file, in 512-byte units.

25840 **OPTIONS**

25841           The *sum* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 25842           following options are supported:

25843 UN       **-r**       Use an alternative algorithm in computing the checksum.

25844 **OPERANDS**

25845           The following operands are supported:

25846       *file*       A pathname of a file. If no files are named, the standard input is read.

25847 **STDIN**

25848           The standard input is any type of file, used only if no *file* operands are specified.

25849 **INPUT FILES**

25850           The input files are of any file type.

25851 **ENVIRONMENT VARIABLES**

25852           The following environment variables may affect the execution of *sum*:

25853       **LANG**       Provide a default value for the internationalisation variables that are unset or null. If  
 25854       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 25855       default locale will be used. If any of the internationalisation variables contains an  
 25856       invalid setting, the utility will behave as if none of the variables had been defined.

25857       **LC\_ALL**

25858           If set to a non-empty string value, override the values of all the other  
 25859           internationalisation variables.

25860       **LC\_CTYPE**

25861           Determine the locale for the interpretation of sequences of bytes of text data as  
 25862           characters (for example, single- as opposed to multi-byte characters in arguments).

25863       **LC\_MESSAGES**

25864           Determine the locale that should be used to affect the format and contents of diagnostic  
 25865           messages written to standard error.

25866       **NLSPATH**

25867           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

25868 **ASYNCHRONOUS EVENTS**

25869           Default.

25870 **STDOUT**

25871           For each file processed successfully, *sum* writes a line of the following format:

25872           "%u %d %s\n", <checksum>, <number of 512-byte units>, <pathname>

25873           If the standard input is used for input, the pathname is omitted.



25874 **STDERR**

25875           Used only for diagnostic messages.

25876 **OUTPUT FILES**

25877           None.

25878 **EXTENDED DESCRIPTION**

25879           None.

25880 **EXIT STATUS**

25881           The following exit values are returned:

25882           0   Successful completion.

25883           &gt;0  An error occurred.

25884 **CONSEQUENCES OF ERRORS**

25885           Default.

25886 **APPLICATION USAGE**

25887           It is not clear that the algorithms used in typical implementations are portable, that is, the same  
 25888           checksum might not be produced for the same input on different systems. Portable applications  
 25889           should use *cksum*.

25890 **EXAMPLES**

25891           None.

25892 **FUTURE DIRECTIONS**

25893           None.

25894 **SEE ALSO**25895           *cksum*.25896 **CHANGE HISTORY**

25897           First released in Issue 2.

25898 **Issue 4**

25899           Format reorganised.

25900           Utility Syntax Guidelines support mandated.

25901           Internationalised environment variable support made optional.

25902           Marked TO BE WITHDRAWN.

25903 **Issue 5**

25904           Marked LEGACY.

25905 **NAME**

25906        **tabs** — set terminal tabs

25907 **SYNOPSIS**

25908 EX UN    **tabs** [ *-n* | *-a* | *-a2* | *-c* | *-c2* | *-c3* | *-f* | *-p* | *-s* | *-u* ] [ *+m[n]* ]  
 25909        [ *-T type* ]

25910 EX        **tabs** [ *-T type* ] [ *+m[n]* ] *n1* [ *,n2* , ... ]

25911 **DESCRIPTION**

25912        The *tabs* utility displays a series of characters that first clears the hardware terminal tab settings  
 25913 EX        and then initialises the tab stops at the specified positions and optionally adjusts the margin.

25914        The phrase “tab-stop position *N*” is taken to mean that, from the start of a line of output,  
 25915        tabbing to position *N* will cause the next character output to be in the (*N*+1)th column position  
 25916        on that line. The maximum number of tab stops allowed is terminal-dependent.

25917        It need not be possible to implement *tabs* on certain terminals. If the terminal type obtained  
 25918        from the *TERM* environment variable or *-T* option represents such a terminal, an appropriate  
 25919        diagnostic message will be written to standard error and *tabs* will exit with a status greater than  
 25920        zero.

25921 **OPTIONS**

25922 EX        The *tabs* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**, except  
 25923        for various extensions: the options *-a2*, *-c2* and *-c3* are multi-character and *+m[n]* uses a  
 25924        leading plus sign and an optional option-argument.

25925        The following options are supported:

25926        *-n*        Specify repetitive tab stops separated by a uniform number of column positions, *n*,  
 25927               where *n* is a single-digit decimal number. The default usage of *tabs* with no arguments  
 25928               is equivalent to *tabs -8*. When *-0* is used, the tab stops are cleared and no new ones set.

25929 EX        *-a*        1,10,16,36,72  
 25930               Assembler, applicable to some mainframes.

25931 EX        *-a2*       1,10,16,40,72  
 25932               Assembler, applicable to some mainframes.

25933 EX        *-c*        1,8,12,16,20,55  
 25934               COBOL, normal format.

25935 EX        *-c2*       1,6,10,14,49  
 25936               COBOL, compact format (columns 1–6 omitted).

25937 EX        *-c3*       1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67  
 25938               COBOL compact format (columns 1–6 omitted), with more tabs than *-c2*.

25939 EX        *-f*        1,7,11,15,19,23  
 25940               FORTRAN

25941 EX        *-p*        1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61  
 25942               PL/1

25943 EX        *-s*        1,10,55  
 25944               SNOBOL

25945 EX **-u** 1,12,20,44  
 25946 Assembler, applicable to some mainframes.

25947 **-T type**  
 25948 Indicate the type of terminal. If this option is not supplied and the *TERM* variable is  
 25949 unset or null, an unspecified default terminal type will be used. The setting of *type* will  
 25950 take precedence over the value in *TERM*.

25951 EX UN **+m [n]**  
 25952 Reset the margin. The margin argument can be used for some terminals. It causes all  
 25953 tabs to be moved over *n* columns by making column *n*+1 the left margin. If *n* is  
 25954 omitted, the default is 10. The normal (leftmost) margin on most terminals is obtained  
 25955 by +m0. The margin for most terminals is reset only when the +m flag is given  
 25956 explicitly.

25957 **OPERANDS**  
 25958 The following operand is supported:

25959 **n1[,n2,...]**  
 25960 A single command-line argument that consists of tab-stop values separated using  
 25961 either commas or blank characters. The tab-stop values will be positive decimal  
 25962 integers in strictly ascending order. If any number (except the first one) is preceded by  
 25963 a plus sign, it is taken as an increment to be added to the previous value. For example,  
 25964 the tab lists 1,10,20,30 and 1,10,+10,+10 are considered to be identical.

25965 **STDIN**  
 25966 Not used.

25967 **INPUT FILES**  
 25968 None.

25969 **ENVIRONMENT VARIABLES**  
 25970 The following environment variables affect the execution of *tabs*:

25971 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 25972 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 25973 default locale will be used. If any of the internationalisation variables contains an  
 25974 invalid setting, the utility will behave as if none of the variables had been defined.

25975 **LC\_ALL**  
 25976 If set to a non-empty string value, override the values of all the other  
 25977 internationalisation variables.

25978 **LC\_CTYPE**  
 25979 Determine the locale for the interpretation of sequences of bytes of text data as  
 25980 characters (for example, single- as opposed to multi-byte characters in arguments).

25981 **LC\_MESSAGES**  
 25982 Determine the locale that should be used to affect the format and contents of diagnostic  
 25983 messages written to standard error.

25984 EX **NLSPATH**  
 25985 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

25986 **TERM** Determine the terminal type. If this variable is unset or null, and if the **-T** option is not  
 25987 specified, an unspecified default terminal type will be used.

25988 **ASYNCHRONOUS EVENTS**  
 25989 Default.

25990 **STDOUT**

25991 If standard output is a terminal, the appropriate sequence to clear and set the tab stops may be  
 25992 written to standard output in an unspecified format. If standard output is not a terminal,  
 25993 undefined results occur.

25994 **STDERR**

25995 Used only for diagnostic messages.

25996 **OUTPUT FILES**

25997 None.

25998 **EXTENDED DESCRIPTION**

25999 None.

26000 **EXIT STATUS**

26001 The following exit values are returned:

26002 0 Successful completion.

26003 >0 An error occurred.

26004 **CONSEQUENCES OF ERRORS**

26005 Default.

26006 **APPLICATION USAGE**

26007 This utility makes use of the terminal's hardware tabs and the *stty tabs* option.

26008 This utility is not recommended for application use.

26009 Some integrated display units might not have escape sequences to set tab stops, but may be set  
 26010 by internal system calls. On these terminals, *tabs* will work if standard output is directed to the  
 26011 terminal; if output is directed to another file, however, *tabs* will fail.

26012 **EXAMPLES**

26013 None.

26014 **FUTURE DIRECTIONS**

26015 None.

26016 **SEE ALSO**

26017 *expand, stty, unexpand.*

26018 **CHANGE HISTORY**

26019 First released in Issue 2.

26020 **Issue 4**

26021 Aligned with the ISO/IEC 9945-2: 1993 standard.

26022 **NAME**

26023           tail — copy the last part of a file

26024 **SYNOPSIS**26025           tail [-f][ -c *number*| -n *number*] [*file*]26026 EX OB   tail -[*number*][b|c|l][f] [*file*]26027 EX OB   tail +[*number*][b|c|l][f] [*file*]26028 **DESCRIPTION**26029           The *tail* utility copies its input file to the standard output beginning at a designated place.

26030 OB       Copying begins at the point in the file indicated by the **-c *number*** or **-n *number*** options (or the  
 26031 ***±number*** portion of the argument to the obsolescent version). The option-argument *number* is  
 26032 EX OB   counted in units of lines or bytes, according to the options **-n** and **-c** (or, in the obsolescent  
 26033 version, the appended option suffixes **l** (lines), **b** (512-byte blocks) or **c** (bytes)). Both line and  
 26034 byte counts start from 1.

26035           Tails relative to the end of the file may be saved in an internal buffer, and thus may be limited in  
 26036 length. Such a buffer, if any, will be no smaller than {LINE\_MAX}\*10 bytes.

26037 **OPTIONS**

26038 OB       The *tail* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 26039 that the obsolescent version accepts multi-character options that can preceded by a plus sign.

26040           The following options are supported:

26041           **-c *number***

26042           The *number* option-argument must be a decimal integer whose sign affects the location  
 26043 in the file, measured in bytes, to begin the copying:  
 26044

26045

| Sign        | Copying Starts                         |
|-------------|----------------------------------------|
| +           | Relative to the beginning of the file. |
| -           | Relative to the end of the file.       |
| <i>none</i> | Relative to the end of the file.       |

26046           The origin for counting is 1; that is, **-c +1** represents the first byte of the file, **-c -1** the  
 26047 last.  
 26048

26051           **-f**       If the input file is a regular file or if the *file* operand specifies a FIFO, do not terminate  
 26052 after the last line of the input file has been copied, but read and copy further bytes from  
 26053 the input file when they become available. If no *file* operand is specified and standard  
 26054 input is a pipe, the **-f** option will be ignored. If the input file is not a FIFO, pipe or  
 26055 regular file, it is unspecified whether or not the **-f** option will be ignored.

26056           **-n *number***

26057           This option is equivalent to **-c *number***, except the starting location in the file is  
 26058 measured in lines instead of bytes. The origin for counting is 1; that is, **-n +1**  
 26059 represents the first line of the file, **-n -1** the last.

26060           In the non-obsolescent form, if neither **-c** nor **-n** is specified, **-n 10** is assumed.

26061 OB In the obsolescent version, an argument beginning with a "-" or "+" can be used as a single  
 26062 option. The argument  $\pm number$  with the letter c specified as a suffix is equivalent to  $-c \pm number$ ;  
 26063  $\pm number$  with the b suffix is equivalent to  $-c \pm number*512$ ;  $\pm number$  with the letter l specified as  
 26064 a suffix, or with none of b, c nor l as a suffix, is equivalent to  $-n \pm number$ . If  $number$  is not  
 26065 specified in these forms, 10 will be used. The letter f specified as a suffix is equivalent to  
 26066 specifying the  $-f$  option. If the  $-[number]c[f]$  form is used and neither  $number$  nor the f suffix  
 26067 is specified, it will be interpreted as the  $-c 10$  option.

## 26068 OPERANDS

26069 The following operand is supported:

26070 *file* A pathname of an input file. If no *file* operands are specified, the standard input will be  
 26071 used.

## 26072 STDIN

26073 The standard input will be used only if no *file* operands are specified. See the INPUT FILES  
 26074 section.

## 26075 INPUT FILES

26076 If the  $-c$  option is specified, the input file can contain arbitrary data; otherwise, the input file  
 26077 must be a text file.

## 26078 ENVIRONMENT VARIABLES

26079 The following environment variables affect the execution of *tail*:

26080 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 26081 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 26082 default locale will be used. If any of the internationalisation variables contains an  
 26083 invalid setting, the utility will behave as if none of the variables had been defined.

26084 *LC\_ALL*

26085 If set to a non-empty string value, override the values of all the other  
 26086 internationalisation variables.

26087 *LC\_CTYPE*

26088 Determine the locale for the interpretation of sequences of bytes of text data as  
 26089 characters (for example, single- as opposed to multi-byte characters in arguments and  
 26090 input files).

26091 *LC\_MESSAGES*

26092 Determine the locale that should be used to affect the format and contents of diagnostic  
 26093 messages written to standard error.

26094 EX *NLSPATH*

26095 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

## 26096 ASYNCHRONOUS EVENTS

26097 Default.

## 26098 STDOUT

26099 The designated portion of the input file will be written to standard output.

## 26100 STDERR

26101 Used only for diagnostic messages.

## 26102 OUTPUT FILES

26103 None.

26104 **EXTENDED DESCRIPTION**

26105       None.

26106 **EXIT STATUS**

26107       The following exit values are returned:

26108           0   Successful completion.

26109           &gt;0  An error occurred.

26110 **CONSEQUENCES OF ERRORS**

26111       Default.

26112 **APPLICATION USAGE**26113       The `-c` option should be used with caution when the input is a text file containing multi-byte characters; it may produce output that does not start on a character boundary.

26115       Although the input file to *tail* can be any type, the results might not be what would be expected on some character special device files or on file types not described by the **XSH** specification. Since this specification does not specify the block size used when doing input, *tail* need not read all of the data from devices that only perform block transfers.

26119 EX OB   The `b` suffix in the obsolescent version is not portable outside of XSI-conformant systems.26120 **EXAMPLES**26121       The `-f` option can be used to monitor the growth of a file that is being written by some other process. For example, the command:26123           `tail -f fred`26124       prints the last ten lines of the file **fred**, followed by any lines that are appended to **fred** between the time **tail** is initiated and killed. As another example, the command:26126           `tail -f -c 15 fred`26127       prints the last 15 bytes of the file **fred**, followed by any bytes that are appended to **fred** between the time **tail** is initiated and killed.26129 **FUTURE DIRECTIONS**

26130       None.

26131 **SEE ALSO**26132       *head*.26133 **CHANGE HISTORY**

26134       First released in Issue 2.

26135 **Issue 4**

26136       Aligned with the ISO/IEC 9945-2: 1993 standard.

## 26137 NAME

26138 talk — talk to another user

## 26139 SYNOPSIS

26140 talk *address* [*terminal*]

## 26141 DESCRIPTION

26142 The *talk* utility is a two-way, screen-oriented communication program.26143 When first invoked, *talk* sends a message similar to:

26144 Message from <*unspecified string*>  
 26145 talk: connection requested by *your\_address*  
 26146 talk: respond with: talk *your\_address*

26147 to the specified *address*. At this point, the recipient of the message can reply by typing:26148 talk *your\_address*

26149 Once communication is established, the two parties can type simultaneously, with their output  
 26150 displayed in separate regions of the screen. Characters are processed as follows:

- 26151 • Typing the alert character will alert the recipient's terminal.
- 26152 • Typing <control>-L will cause the sender's screen regions to be refreshed.
- 26153 • Typing the erase and kill characters will affect the sender's terminal in the manner described  
 26154 by the **termios** interface in the **XBD** specification, **Chapter 9, General Terminal Interface**.
- 26155 • Typing the interrupt or end-of-file characters will terminate the local *talk* utility. Once the  
 26156 *talk* session has been terminated on one side, the other side of the *talk* session will be notified  
 26157 that the *talk* session has been terminated and will be able to do nothing except exit.
- 26158 • Typing characters from LC\_CTYPE classifications **print** or **space** will cause those characters  
 26159 to be sent to the recipient's terminal.
- 26160 • When and only when the *stty ixten* local mode is enabled, the existence and processing of  
 26161 additional special control characters and multi-byte or single-byte functions is  
 26162 implementation-dependent.
- 26163 • Typing other non-printable characters will cause implementation-dependent sequences of  
 26164 printable characters to be sent to the recipient's terminal.

26165 Permission to be a recipient of a *talk* message can be denied or granted by use of the *mesg* utility.  
 26166 However, a user's privilege may further constrain the domain of accessibility of other users'  
 26167 terminals. The *talk* utility will fail when the user lacks the appropriate privileges to perform the  
 26168 requested action.

26169 Certain block-mode terminals do not have all the capabilities necessary to support the  
 26170 simultaneous exchange of messages required for *talk*. When this type of exchange cannot be  
 26171 supported on such terminals, the implementation may support an exchange with reduced levels  
 26172 of simultaneous interaction or it may report an error describing the terminal-related deficiency.

## 26173 OPTIONS

26174 None.

## 26175 OPERANDS

26176 The following operands are supported:

26177 *address* The recipient of the *talk* session. One form of *address* is the <*user name*>, as returned by  
 26178 the *who* utility. Other address formats and how they are handled are unspecified.



26179 *terminal*

26180 If the recipient is logged in more than once, the *terminal* argument can be used to  
 26181 indicate the appropriate terminal name. If *terminal* is not specified, the *talk* message  
 26182 will be displayed on one or more accessible terminals in use by the recipient. The  
 26183 format of *terminal* will be the same as that returned by the *who* utility.

#### 26184 **STDIN**

26185 Characters read from standard input will be copied to the recipient's terminal in an unspecified  
 26186 manner. If standard input is not a terminal, *talk* will write a diagnostic message and exit with a  
 26187 non-zero status.

#### 26188 **INPUT FILES**

26189 None.

#### 26190 **ENVIRONMENT VARIABLES**

26191 The following environment variables affect the execution of *talk*:

26192 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 26193 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 26194 default locale will be used. If any of the internationalisation variables contains an  
 26195 invalid setting, the utility will behave as if none of the variables had been defined.

26196 *LC\_ALL*

26197 If set to a non-empty string value, override the values of all the other  
 26198 internationalisation variables.

26199 *LC\_CTYPE*

26200 Determine the locale for the interpretation of sequences of bytes of text data as  
 26201 characters (for example, single- as opposed to multi-byte characters in arguments and  
 26202 input files). If the recipient's locale does not use an *LC\_CTYPE* equivalent to the  
 26203 sender's, the results are undefined.

26204 *LC\_MESSAGES*

26205 Determine the locale that should be used to affect the format and contents of diagnostic  
 26206 messages written to standard error and informative messages written to standard  
 26207 output.

26208 EX *NLSPATH*

26209 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

26210 *TERM* Determine the name of the invoker's terminal type. If this variable is unset or null, an  
 26211 unspecified default terminal type will be used.

#### 26212 **ASYNCHRONOUS EVENTS**

26213 When the *talk* utility receives a SIGINT signal, the utility will terminate and exit with a zero  
 26214 status. It will take the standard action for all other signals.

#### 26215 **STDOUT**

26216 If standard output is a terminal, characters copied from the recipient's standard input may be  
 26217 written to standard output. Standard output also may be used for diagnostic messages. If  
 26218 standard output is not a terminal, *talk* will exit with a non-zero status.

#### 26219 **STDERR**

26220 None.

#### 26221 **OUTPUT FILES**

26222 None.

**26223 EXTENDED DESCRIPTION**

26224           None.

**26225 EXIT STATUS**

26226           The following exit values are returned:

26227           0   Successful completion.

26228           >0  An error occurred or *talk* was invoked on a terminal incapable of supporting it.

**26229 CONSEQUENCES OF ERRORS**

26230           Default.

**26231 APPLICATION USAGE**

26232           Because the handling of non-printable, non-space characters is tied to the *stty* description of  
26233           *iexten*, implementation extensions within the terminal driver can be accessed. For example,  
26234           some implementations provide line editing functions with certain control character sequences.

**26235 EXAMPLES**

26236           None.

**26237 FUTURE DIRECTIONS**

26238           None.

**26239 SEE ALSO**

26240           *mesg*, *who*, *write*, the **XBD** specification, **Chapter 9, General Terminal Interface**.

**26241 CHANGE HISTORY**

26242           First released in Issue 4.

26243 **NAME**26244 tar — file archiver (**LEGACY**)26245 **SYNOPSIS**26246 EX tar key [*file...*]26247 **DESCRIPTION**26248 The *tar* utility processes archives of files. Its actions are controlled by the *key* operand.26249 **OPTIONS**

26250 None.

26251 **OPERANDS**

26252 The following operands are supported:

26253 *key* The *key* operand consists of a function letter followed immediately by zero or more  
 26254 modifying letters.

26255 The function letter is one of the following:

26256 **r** Write the named *file* or files on the end of the archive. If the archive is on a  
 26257 magnetic tape device, the results are unspecified.

26258 **x** Extract the named *file* or files from the archive. If a named file matches a  
 26259 directory whose contents had been written onto the archive, this directory is  
 26260 (recursively) extracted. If a named file in the archive does not exist on the  
 26261 system, the file is created with the same mode as the one in the archive, except  
 26262 that the set-user-ID and set-group-ID modes are not set unless the user has  
 26263 appropriate privileges. If the files exist, their modes are not changed except as  
 26264 described above. The owner, group, and modification time are restored (if  
 26265 possible). If no *file* operand is given, the entire content of the archive is  
 26266 extracted. Note that if several files with the same name are in the archive, the  
 26267 last one overwrites all earlier ones.

26268 **t** Write to standard output the names of all the files in the archive.

26269 **u** Add the named *file* or files to the archive if they are not already there, or have  
 26270 been modified since last written into the archive. If the archive is on a  
 26271 magnetic tape device, the results are unspecified.

26272 **c** Create a new archive; writing begins at the beginning of the archive, instead of  
 26273 after the last file.

26274 The following characters can be appended to the function letter. Appending the same  
 26275 character more than once produces undefined results. The order of the b and f  
 26276 characters is significant.

26277 **v** (Verbose.) Write to standard error the name of each file processed, preceded  
 26278 by a string indicating the operation being performed, as follows:

26279

26280

26281

26282

| Key Letter | String |
|------------|--------|
| c, r, u    | "a "   |
| x          | "x "   |

26283 The filename may be followed by additional information, such as the size of  
 26284 the file in the archive or file system, in an unspecified format. When used  
 26285 with the t function letter, v writes to standard output more information about  
 26286 the archive entries than just the name.

|       |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26287 | <b>w</b>                     | Write the action to be taken, followed by the name of the file, and then wait for the user's confirmation. If an affirmative response is given, the action is performed. Any other input suppresses the action.                                                                                                                                                                                                                                                                                                                       |
| 26288 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26289 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26290 | <b>f</b>                     | Use the first <i>file</i> operand (or the second, if <b>b</b> has already been specified) as the name of the archive instead of the system-dependent default. If the name of the file is <b>-</b> , <i>tar</i> writes to the standard output or reads from the standard input, whichever is appropriate. Thus, <i>tar</i> can be used as the head or tail of a pipeline. The <i>tar</i> utility can also be used to move directory hierarchies with the command:                                                                      |
| 26291 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26292 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26293 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26294 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26295 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26296 |                              | <code>(cd fromdir; tar cf - . )   (cd todir; tar xf -)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 26297 | <b>b</b>                     | Use the first <i>file</i> operand (or the second, if <b>f</b> has already been specified) as the blocking factor for tape records. The default is not greater than 20; the maximum is not less than 20. This modifier should only be used with raw magnetic tape archives (see <b>f</b> above). The block size is determined automatically when reading tapes (function letters <b>x</b> and <b>t</b> ).                                                                                                                              |
| 26298 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26299 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26300 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26301 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26302 | <b>l</b>                     | Report if all of the links to the files being archived cannot be resolved. If <b>l</b> is not specified, no error messages are written.                                                                                                                                                                                                                                                                                                                                                                                               |
| 26303 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26304 | <b>m</b>                     | Do not restore the modification times. The modification time of the file will be the time of extraction.                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 26305 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26306 | <b>o</b>                     | Assign to extracted files the user and group identifier of the user running the program rather than those on the archive.                                                                                                                                                                                                                                                                                                                                                                                                             |
| 26307 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26308 | <i>file</i>                  | A pathname of a regular file or directory to be archived (when the <b>c</b> , <b>r</b> or <b>u</b> function letters are used), extracted ( <b>x</b> ) or listed ( <b>t</b> ). When <i>file</i> is the pathname of a directory, the action applies to all of the files and (recursively) subdirectories of that directory. When either or both of the <b>b</b> or <b>f</b> letters are used in the <i>key</i> operand, the initial <i>file</i> operands are interpreted as a blocking factor or archive name, as described previously. |
| 26309 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26310 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26311 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26312 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26313 | <b>STDIN</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26314 |                              | When the <b>f</b> modifier is used with the <b>t</b> or <b>x</b> function letter and the pathname is <b>-</b> , the standard input is an archive file formatted as specified by <i>pax</i> with the <b>-x ustar</b> option. Otherwise, the standard input is not used.                                                                                                                                                                                                                                                                |
| 26315 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26316 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26317 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26318 |                              | The files identified by the <i>file</i> operands are regular files or directories.                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 26319 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26320 |                              | The following environment variables may affect the execution of <i>tar</i> :                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 26321 | <b>LANG</b>                  | Provide a default value for the internationalisation variables that are unset or null. If <b>LANG</b> is unset or null, the corresponding value from the implementation-dependent default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.                                                                                                                                                                        |
| 26322 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26323 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26324 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26325 | <b>LC_ALL</b>                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26326 |                              | If set to a non-empty string value, override the values of all the other internationalisation variables.                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 26327 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26328 | <b>LC_COLLATE</b>            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26329 |                              | Determine the locale for the behaviour of ranges, equivalence classes and multi-character collating elements used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <b>LC_MESSAGES</b> category.                                                                                                                                                                                                                                                                                                |
| 26330 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26331 |                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

26332 **LC\_CTYPE**

26333 Determine the locale for the interpretation of sequences of bytes of text data as  
 26334 characters (for example, single- as opposed to multi-byte characters in arguments and  
 26335 input files) and the behaviour of character classes used in the extended regular  
 26336 expression defined for the **yesexpr** locale keyword in the LC\_MESSAGES category.

26337 **LC\_MESSAGES**

26338 Determine the locale for the processing of affirmative responses and that should be  
 26339 used to affect the format and contents of diagnostic messages written to standard error.

26340 **LC\_TIME**

26341 Determine the format of date and time strings output when listing the contents of an  
 26342 archive with the **v** modifier; for example:

26343 `tar tvf /dev/tape`

26344 **NLSPATH**

26345 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

26346 **TZ** Determine the timezone used with date and time strings.

26347 **ASYNCHRONOUS EVENTS**

26348 Default.

26349 **STDOUT**

26350 When the **f** modifier is used with the **r**, **u** or **c** function letter and the pathname is **-**, the standard  
 26351 output is an archive file formatted as specified by *pax* with the **-x ustar** option. When the **t**  
 26352 function letter is used, the standard output consists of the names of the files in the archive,  
 26353 separated by newline characters; if **v** is used with **t**, the standard output includes additional  
 26354 information in an unspecified format. Otherwise, the standard output is not used.

26355 **STDERR**

26356 The standard error is used for diagnostic messages and the filename output described under the  
 26357 **v** modifier (when the **t** function letter is not used).

26358 **OUTPUT FILES**

26359 Output files are created, as specified by the archive, when the **x** function letter is used.

26360 **EXTENDED DESCRIPTION**

26361 None.

26362 **EXIT STATUS**

26363 The following exit values are returned:

26364 **0** Successful completion.

26365 **>0** An error occurred.

26366 **CONSEQUENCES OF ERRORS**

26367 Default.

26368 **APPLICATION USAGE**

26369 Some systems have usually had blocking factors in the range 1 to at least 127 with a default of 20  
 26370 while other systems have usually had blocking factors in the range 1 to 20 with a default of 1.  
 26371 For maximum portability, applications should specify a blocking factor no larger than 20.

26372 For portable communication of data between XSI-conformant systems, it is recommended that  
26373 only characters defined in the ISO/IEC 646:1991 standard International Reference Version  
26374 (equivalent to ASCII) 7-bit range of characters be used and that only characters defined in the  
26375 Portable Filename Character Set be used for naming files. This recommendation is given  
26376 because XSI-conformant systems support diverse codesets and run in various geographical areas  
26377 and there is no single, well established codeset that incorporates all of the characters of the  
26378 languages of the various geographical areas.

26379 Note that the *tar* format can only support files up to 8 gigabytes in size.

26380 Applications should migrate to the *pax* utility.

26381 **EXAMPLES**

26382 None.

26383 **FUTURE DIRECTIONS**

26384 None.

26385 **SEE ALSO**

26386 *cpio*, *pax*.

26387 **CHANGE HISTORY**

26388 First released in Issue 2.

26389 **Issue 4**

26390 Format reorganised.

26391 Marked TO BE WITHDRAWN.

26392 Internationalised environment variable support made optional.

26393 **Issue 5**

26394 A note is added to the APPLICATION USAGE section indicating that the *tar* format can only  
26395 support files up to 8 gigabytes in size.

26396 Marked LEGACY.

26397 **NAME**

26398       tee — duplicate standard input

26399 **SYNOPSIS**26400       tee [-ai][*file...*]26401 **DESCRIPTION**26402       The *tee* utility will copy standard input to standard output, making a copy in zero or more files.26403       The *tee* utility will not buffer output.

26404       The options determine if the specified files are overwritten or appended to.

26405 **OPTIONS**26406       The *tee* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

26407       The following options are supported:

26408       **-a**       Append the output to the files rather than overwriting them.26409       **-i**       Ignore the SIGINT signal.26410 **OPERANDS**

26411       The following operands are supported:

26412       *file*       A pathname of an output file. Processing of at least 13 *file* operands will be supported.26413 **STDIN**

26414       The standard input can be of any type.

26415 **INPUT FILES**

26416       None.

26417 **ENVIRONMENT VARIABLES**26418       The following environment variables affect the execution of *tee*:

26419       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 26420       **LANG** is unset or null, the corresponding value from the implementation-dependent  
 26421       default locale will be used. If any of the internationalisation variables contains an  
 26422       invalid setting, the utility will behave as if none of the variables had been defined.

26423       **LC\_ALL**

26424       If set to a non-empty string value, override the values of all the other  
 26425       internationalisation variables.

26426       **LC\_CTYPE**

26427       Determine the locale for the interpretation of sequences of bytes of text data as  
 26428       characters (for example, single- as opposed to multi-byte characters in arguments).

26429       **LC\_MESSAGES**

26430       Determine the locale that should be used to affect the format and contents of diagnostic  
 26431       messages written to standard error.

26432 EX       **NLSPATH**26433       Determine the location of message catalogues for the processing of **LC\_MESSAGES**.26434 **ASYNCHRONOUS EVENTS**26435       Default, except that if the **-i** option was specified, SIGINT will be ignored.26436 **STDOUT**

26437       The standard output will be a copy of the standard input.

26438 **STDERR**

26439       Used only for diagnostic messages.

26440 **OUTPUT FILES**26441       If any *file* operands are specified, the standard input will be copied to each named file.26442 **EXTENDED DESCRIPTION**

26443       None.

26444 **EXIT STATUS**

26445       The following exit values are returned:

26446           0   The standard input was successfully copied to all output files.

26447           &gt;0  An error occurred.

26448 **CONSEQUENCES OF ERRORS**

26449       If a write to any successfully opened *file* operand fails, writes to other successfully opened *file*  
26450       operands and standard output will continue, but the exit status will be non-zero. Otherwise, the  
26451       default actions specified in Section 1.9 on page 11 will apply.

26452 **APPLICATION USAGE**26453       The *tee* utility is usually used in a pipeline, to make a copy of the output of some utility.26454       The *file* operand is technically optional, but *tee* is no more useful than *cat* when none is specified.26455 **EXAMPLES**

26456       Save an unsorted intermediate form of the data in a pipeline:

26457           ... | tee unsorted | sort &gt; sorted

26458 **FUTURE DIRECTIONS**

26459       None.

26460 **SEE ALSO**26461       *cat*.26462 **CHANGE HISTORY**

26463       First released in Issue 2.

26464 **Issue 4**

26465       Aligned with the ISO/IEC 9945-2: 1993 standard.



26466 **NAME**

26467           test — evaluate expression

26468 **SYNOPSIS**26469           test [*expression*]26470           [ [*expression*] ]26471 **DESCRIPTION**

26472       The *test* utility evaluates the *expression* and indicates the result of the evaluation by its exit status.  
 26473       An exit status of zero indicates that the expression evaluated as true and an exit status of 1  
 26474       indicates that the expression evaluated as false.

26475       In the second form of the utility, which uses [] rather than *test*, the square brackets must be  
 26476       separate arguments.

26477 **OPTIONS**

26478       The *test* utility does not recognise the -- argument in the manner specified by guideline 10 in  
 26479       the XBD specification, **Section 10.2, Utility Syntax Guidelines**.

26480       No options are supported.

26481 **OPERANDS**

26482       All operators and elements of primaries must be presented as separate arguments to the *test*  
 26483       utility.

26484       The following primaries can be used to construct *expression*:26485       **-b file**   True if *file* exists and is a block special file.26486       **-c file**   True if *file* exists and is a character special file.26487       **-d file**   True if *file* exists and is a directory.26488       **-e file**   True if *file* exists.26489       **-f file**   True if *file* exists and is a regular file.26490       **-g file**   True if *file* exists and its set group ID flag is set.26491       **-n string**26492           True if the length of *string* is non-zero.26493       **-p file**   True if *file* is a named pipe (FIFO).26494       **-r file**   True if *file* exists and is readable.26495       **-s file**   True if *file* exists and has a size greater than zero.26496       **-t file\_descriptor**

26497           True if the file whose file descriptor number is *file\_descriptor* is open and is associated  
 26498           with a terminal.

26499       **-u file**   True if *file* exists and its set-user-ID flag is set.

26500       **-w file**   True if *file* exists and is writable. True will indicate only that the write flag is on. The  
 26501       *file* will not be writable on a read-only file system even if this test indicates true.

26502       **-x file**   True if *file* exists and is executable. True will indicate only that the execute flag is on. If  
 26503       *file* is a directory, true indicates that *file* can be searched.

26504        *-z string*  
 26505            True if the length of string *string* is zero.

26506        *string*    True if the string *string* is not the null string.

26507        *s1 = s2*    True if the strings *s1* and *s2* are identical.

26508        *s1 != s2*   True if the strings *s1* and *s2* are not identical.

26509        *n1 -eq n2*  
 26510            True if the integers *n1* and *n2* are algebraically equal.

26511        *n1 -ne n2*  
 26512            True if the integers *n1* and *n2* are not algebraically equal.

26513        *n1 -gt n2*  
 26514            True if the integer *n1* is algebraically greater than the integer *n2*.

26515        *n1 -ge n2*  
 26516            True if the integer *n1* is algebraically greater than or equal to the integer *n2*.

26517        *n1 -lt n2*  
 26518            True if the integer *n1* is algebraically less than the integer *n2*.

26519        *n1 -le n2*  
 26520            True if the integer *n1* is algebraically less than or equal to the integer *n2*.

26521 EX     *expression1 -a expression2*  
 26522            True if both *expression1* and *expression2* are true. The *-a* binary primary is left  
 26523            associative. It has a higher precedence than *-o*.

26524 EX     *expression1 -o expression2*  
 26525            True if either *expression1* or *expression2* is true. The *-o* binary primary is left associative.

26526        These primaries can be combined with the following operators:

26527        *! expression*  
 26528            True if *expression* is false.

26529 EX     *( expression )*  
 26530            True if *expression* is true. The parentheses can be used to alter the normal precedence  
 26531            and associativity.

26532        The primaries with two elements of the form:

26533            *-primary\_operator primary\_operand*

26534        are known as *unary primaries*. The primaries with three elements in either of the two forms:

26535            *primary\_operand -primary\_operator primary\_operand*

26536            *primary\_operand primary\_operator primary\_operand*

26537        are known as *binary primaries*. Additional implementation-dependent operators and  
 26538        *primary\_operators* may be provided by implementations. They will be of the form *-operator*  
 26539        where the first character of *operator* is not a digit.

26540        The algorithm for determining the precedence of the operators and the return value that will be  
 26541        generated is based on the number of arguments presented to *test*. (However, when using the  
 26542        [...] form, the right-bracket final argument will not be counted in this algorithm.)

26543        In the following list, \$1, \$2, \$3 and \$4 represent the arguments presented to *test*.

|          |                              |                                                                                           |  |
|----------|------------------------------|-------------------------------------------------------------------------------------------|--|
| 26544    | 0 arguments:                 | Exit false (1).                                                                           |  |
| 26545    | 1 argument:                  | Exit true (0) if \$1 is not null; otherwise, exit false.                                  |  |
| 26546    | 2 arguments:                 |                                                                                           |  |
| 26547    |                              | • If \$1 is "!", exit true if \$2 is null, false if \$2 is not null.                      |  |
| 26548    |                              | • If \$1 is a unary primary, exit true if the unary test is true, false if the unary      |  |
| 26549    |                              | test is false.                                                                            |  |
| 26550    |                              | • Otherwise, produce unspecified results.                                                 |  |
| 26551    | 3 arguments:                 |                                                                                           |  |
| 26552    |                              | • If \$2 is a binary primary, perform the binary test of \$1 and \$3.                     |  |
| 26553    |                              | • If \$1 is "!", negate the two-argument test of \$2 and \$3.                             |  |
| 26554 EX |                              | • If \$1 is "(" and \$3 is ")", perform the unary test of \$2.                            |  |
| 26555    |                              | • Otherwise, produce unspecified results.                                                 |  |
| 26556    | 4 arguments:                 |                                                                                           |  |
| 26557    |                              | • If \$1 is "!", negate the three-argument test of \$2, \$3 and \$4.                      |  |
| 26558 EX |                              | • If \$1 is "(" and \$4 is ")", perform the two-argument test of \$2 and \$3.             |  |
| 26559    |                              | • Otherwise, the results are unspecified.                                                 |  |
| 26560 EX | ">4 arguments:"              | Combinations of primaries and operators are evaluated using the precedence                |  |
| 26561    |                              | and associativity rules described previously. In addition, the string                     |  |
| 26562    |                              | comparison binary primaries "=" and "!=" have a higher precedence than any                |  |
| 26563    |                              | unary primary.                                                                            |  |
| 26564    | <b>STDIN</b>                 |                                                                                           |  |
| 26565    |                              | Not used.                                                                                 |  |
| 26566    | <b>INPUT FILES</b>           |                                                                                           |  |
| 26567    |                              | None.                                                                                     |  |
| 26568    | <b>ENVIRONMENT VARIABLES</b> |                                                                                           |  |
| 26569    |                              | The following environment variables affect the execution of <i>test</i> :                 |  |
| 26570    | <b>LANG</b>                  | Provide a default value for the internationalisation variables that are unset or null. If |  |
| 26571    |                              | <i>LANG</i> is unset or null, the corresponding value from the implementation-dependent   |  |
| 26572    |                              | default locale will be used. If any of the internationalisation variables contains an     |  |
| 26573    |                              | invalid setting, the utility will behave as if none of the variables had been defined.    |  |
| 26574    | <b>LC_ALL</b>                |                                                                                           |  |
| 26575    |                              | If set to a non-empty string value, override the values of all the other                  |  |
| 26576    |                              | internationalisation variables.                                                           |  |
| 26577    | <b>LC_CTYPE</b>              |                                                                                           |  |
| 26578    |                              | Determine the locale for the interpretation of sequences of bytes of text data as         |  |
| 26579    |                              | characters (for example, single- as opposed to multi-byte characters in arguments).       |  |
| 26580    | <b>LC_MESSAGES</b>           |                                                                                           |  |
| 26581    |                              | Determine the locale that should be used to affect the format and contents of diagnostic  |  |
| 26582    |                              | messages written to standard error.                                                       |  |
| 26583 EX | <b>NLSPATH</b>               |                                                                                           |  |
| 26584    |                              | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .   |  |

26585 **ASYNCHRONOUS EVENTS**

26586 Default.

26587 **STDOUT**

26588 Not used.

26589 **STDERR**

26590 Used only for diagnostic messages.

26591 **OUTPUT FILES**

26592 None.

26593 **EXTENDED DESCRIPTION**

26594 None.

26595 **EXIT STATUS**

26596 The following exit values are returned:

26597 0 *expression* evaluated to true.26598 1 *expression* evaluated to false or *expression* was missing.

26599 &gt;1 An error occurred.

26600 **CONSEQUENCES OF ERRORS**

26601 Default.

26602 **APPLICATION USAGE**

26603 Scripts should be careful when dealing with user-supplied input that could be confused with  
 26604 primaries and operators. Unless the application writer knows all the cases that produce input to  
 26605 the script, invocations like:

26606 `test "$1" -a "$2"`

26607 should be written as:

26608 `test "$1" && test "$2"`

26609 to avoid problems if a user supplied values such as \$1 set to "!" and \$2 set to the null string. |  
 26610 That is, in cases where maximal portability is of concern, replace:

26611 `test expr1 -a expr2`

26612 with:

26613 `test expr1 && test expr2`

26614 and replace:

26615 `test expr1 -o expr2`

26616 with:

26617 `test expr1 || test expr2`

26618 but note that, in *test*, **-a** has higher precedence than **-o** while **&&** and **||** have equal precedence  
 26619 in the shell.

26620 Parentheses or braces can be used in the shell command language to effect grouping.

26621 Parentheses must be escaped when using *sh*; for example:26622 `test \( expr1 -a expr2 \) -o expr3`

26623 This command is not always portable outside XSI-conformant systems. The following form can  
 26624 be used instead:

```
26625 (test expr1 && test expr2) || test expr3
```

26626 The two commands:

```
26627 test "$1"
26628 test ! "$1"
```

26629 could not be used reliably on some historical systems. Unexpected results would occur if such a  
 26630 *string* expression were used and \$1 expanded to "!", "(" or a known unary primary. Better  
 26631 constructs are:

```
26632 test -n "$1"
26633 test -z "$1"
```

26634 respectively.

26635 Historical systems have also been unreliable given the common construct:

```
26636 test "$response" = "expected string"
```

26637 One of the following is a more reliable form:

```
26638 test "X$response" = "Xexpected string"
26639 test "expected string" = "$response"
```

26640 Note that the second form assumes that expected string could not be confused with any unary  
 26641 primary. If expected string starts with "-", "(", "!" or even "=", the first form should be used  
 26642 instead. Using the preceding rules without the marked extensions, any of the three comparison  
 26643 forms is reliable, given any input. (However, note that the strings are quoted in all cases.)

26644 Because the string comparison binary primaries, "=" and !=, have a higher precedence than any  
 26645 unary primary in the greater than 4 argument case, unexpected results can occur if arguments  
 26646 are not properly prepared. For example, in:

```
26647 test -d $1 -o -d $2
```

26648 If \$1 evaluates to a possible directory name of =, the first three arguments are considered a string  
 26649 comparison, which causes a syntax error when the second -d is encountered. One of the  
 26650 following forms prevents this; the second is preferred:

```
26651 test \(-d "$1" \) -o \(-d "$2" \)
26652 test -d "$1" || test -d "$2"
```

26653 Also in the greater than 4 argument case:

```
26654 test "$1" = "bat" -a "$2" = "ball"
```

26655 Syntax errors will occur if \$1 evaluates to "(" or "!". One of the following forms prevents this; the  
 26656 third is preferred:

```
26657 test "X$1" = "Xbat" -a "X$2" = "Xball"
26658 test "$1" = "bat" && test "$2" = "ball"
26659 test "X$1" = "Xbat" && test "X$2" = "Xball"
```

26660 **EXAMPLES**

26661 EX 1. Exit if there are not two or three arguments (two variations):

```
26662 if [$# -ne 2 -a $# -ne 3]; then exit 1; fi
26663 if [$# -lt 2 -o $# -gt 3]; then exit 1; fi
```

26664 2. Perform a *mkdir* if a directory does not exist:

```
26665 test ! -d tempdir && mkdir tempdir
```

26666 3. Wait for a file to become non-readable:

```
26667 while test -r thefile
26668 do
26669 sleep 30
26670 done
26671 echo '"thefile" is no longer readable'
```

26672 4. Perform a command if the argument is one of three strings (two variations):

```
26673 if ["$1" = "pear"] || ["$1" = "grape"] || ["$1" = "apple"]
26674 then
26675 command
26676 fi
26677
26678 case "$1" in
26679 pear|grape|apple) command ;;
26679 esac
```

26680 **FUTURE DIRECTIONS**

26681 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 26682 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 26683 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 26684 finalised.

26685 **SEE ALSO**26686 *find*.26687 **CHANGE HISTORY**

26688 First released in Issue 2.

26689 **Issue 4**

26690 Aligned with the ISO/IEC 9945-2: 1993 standard.

26691 **Issue 5**

26692 FUTURE DIRECTIONS section added.

26693 **NAME**

26694       time — time a simple command

26695 **SYNOPSIS**26696       time [-p] *utility* [*argument...*]26697 **DESCRIPTION**

26698       The *time* utility invokes the utility named by the *utility* operand with arguments supplied as the  
 26699 *argument* operands and writes a message to standard error that lists timing statistics for the  
 26700 utility. The message includes the following information:

- 26701       • The elapsed (real) time between invocation of *utility* and its termination.
- 26702       • The User CPU time, equivalent to the sum of the *tms\_utime* and *tms\_cutime* fields returned by  
 26703 the **XSH** specification *times()* function for the process in which *utility* is executed.
- 26704       • The System CPU time, equivalent to the sum of the *tms\_stime* and *tms\_cstime* fields returned  
 26705 by the *times()* function for the process in which *utility* is executed.

26706       The precision of the timing will be no less than the granularity defined for the size of the clock  
 26707 tick unit on the system, but the results will be reported in terms of standard time units (for  
 26708 example, 0.02 seconds, 00:00:00.02, 1m33.75s, 365.21 seconds), not numbers of clock ticks.

26709       When *time* is used as part of a pipeline, the times reported are unspecified, except when it is the  
 26710 sole command within a grouping command (see Section 2.9.4 on page 52) in that pipeline. For  
 26711 example, the commands on the left are unspecified; those on the right report on utilities *a* and *c*,  
 26712 respectively:

```
26713 time a | b | c { time a } | b | c
26714 a | b | time c a | b | (time c)
```

26715 **OPTIONS**26716       The *time* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

26717       The following option is supported:

26718       **-p**       Write the timing output to standard error in the format shown in the **STDERR** section.

26719 **OPERANDS**

26720       The following operands are supported:

26721       *utility*   The name of a utility that is to be invoked. If the *utility* operand names any of the  
 26722 special built-in utilities in Section 2.14 on page 67, the results are undefined.

26723       *argument*

26724       Any string to be supplied as an argument when invoking the utility named by the  
 26725 *utility* operand.

26726 **STDIN**

26727       Not used.

26728 **INPUT FILES**

26729       None.

26730 **ENVIRONMENT VARIABLES**26731       The following environment variables affect the execution of *time*:

26732       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 26733 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 26734 default locale will be used. If any of the internationalisation variables contains an  
 26735 invalid setting, the utility will behave as if none of the variables had been defined.

|          |                               |                                                                                                                                                     |
|----------|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 26736    | <b>LC_ALL</b>                 |                                                                                                                                                     |
| 26737    |                               | If set to a non-empty string value, override the values of all the other                                                                            |
| 26738    |                               | internationalisation variables.                                                                                                                     |
| 26739    | <b>LC_CTYPE</b>               |                                                                                                                                                     |
| 26740    |                               | Determine the locale for the interpretation of sequences of bytes of text data as                                                                   |
| 26741    |                               | characters (for example, single- as opposed to multi-byte characters in arguments).                                                                 |
| 26742    | <b>LC_MESSAGES</b>            |                                                                                                                                                     |
| 26743    |                               | Determine the locale that should be used to affect the format and contents of diagnostic                                                            |
| 26744    |                               | and informative messages written to standard error.                                                                                                 |
| 26745    | <b>LC_NUMERIC</b>             |                                                                                                                                                     |
| 26746    |                               | Determine the locale for numeric formatting.                                                                                                        |
| 26747 EX | <b>NLSPATH</b>                |                                                                                                                                                     |
| 26748    |                               | Determine the location of message catalogues for the processing of <b>LC_MESSAGES</b> .                                                             |
| 26749    | <b>PATH</b>                   | Determine the search path that will be used to locate the utility to be invoked. See the                                                            |
| 26750    |                               | <b>XBD</b> specification, <b>Chapter 6, Environment Variables</b> .                                                                                 |
| 26751    | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                     |
| 26752    |                               | Default.                                                                                                                                            |
| 26753    | <b>STDOUT</b>                 |                                                                                                                                                     |
| 26754    |                               | Not used.                                                                                                                                           |
| 26755    | <b>STDERR</b>                 |                                                                                                                                                     |
| 26756    |                               | The standard error will be used to write the timing statistics. If <b>-p</b> is specified, the following                                            |
| 26757    |                               | format will be used in the POSIX locale:                                                                                                            |
| 26758    |                               | "real %f\nuser %f\nsys %f\n", <real seconds>, <user seconds> ,                                                                                      |
| 26759    |                               | <system seconds>                                                                                                                                    |
| 26760    |                               | where each floating-point number is expressed in seconds. The precision used may be less than                                                       |
| 26761    |                               | the default six digits of %f, but will be sufficiently precise to accommodate the size of the clock                                                 |
| 26762    |                               | tick on the system (for example, if there were 60 clock ticks per second, at least two digits follow                                                |
| 26763    |                               | the radix character). The number of digits following the radix character will be no less than one,                                                  |
| 26764    |                               | even if this always results in a trailing zero. The implementation may append white space and                                                       |
| 26765    |                               | additional information following the format shown here.                                                                                             |
| 26766    | <b>OUTPUT FILES</b>           |                                                                                                                                                     |
| 26767    |                               | None.                                                                                                                                               |
| 26768    | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                     |
| 26769    |                               | None.                                                                                                                                               |
| 26770    | <b>EXIT STATUS</b>            |                                                                                                                                                     |
| 26771    |                               | If the <i>utility</i> utility is invoked, the exit status of <i>time</i> will be the exit status of <i>utility</i> ; otherwise,                     |
| 26772    |                               | the <i>time</i> utility will exit with one of the following values: The following exit values are returned:                                         |
| 26773    | 1–125                         | An error occurred in the <i>time</i> utility.                                                                                                       |
| 26774    | 126                           | The utility specified by <i>utility</i> was found but could not be invoked.                                                                         |
| 26775    | 127                           | The utility specified by <i>utility</i> could not be found.                                                                                         |
| 26776    | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                     |
| 26777    |                               | Default.                                                                                                                                            |
| 26778    | <b>APPLICATION USAGE</b>      |                                                                                                                                                     |
| 26779    |                               | The <i>command</i> , <i>env</i> , <i>nice</i> , <i>nohup</i> , <i>time</i> , and <i>xargs</i> utilities have been specified to use exit code 127 if |



26780 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
26781 utility exited with an error indication.” The value 127 was chosen because it is not commonly  
26782 used for other meanings; most utilities use small values for “normal error conditions” and the  
26783 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
26784 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
26785 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
26786 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
26787 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
26788 any other reason.

#### 26789 **EXAMPLES**

26790 It is frequently desirable to apply *time* to pipelines or lists of commands. This can be done by  
26791 placing pipelines and command lists in a single file; this file can then be invoked as a utility, and  
26792 the *time* applies to everything in the file.

26793 Alternatively, the following command can be used to apply *time* to a complex command:

```
26794 time sh -c 'complex-command-line'
```

#### 26795 **FUTURE DIRECTIONS**

26796 None.

#### 26797 **SEE ALSO**

26798 *sh times* special built-in utility.

#### 26799 **CHANGE HISTORY**

26800 First released in Issue 2.

#### 26801 **Issue 4**

26802 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 26803 NAME

26804 touch — change file access and modification times

## 26805 SYNOPSIS

26806 touch [-acm][ -r *ref\_file*| -t *time*] *file*...26807 OB touch [-acm][*date\_time*] *file*...

## 26808 DESCRIPTION

26809 The *touch* utility will change the modification times, access times or both of files. The  
 26810 modification time is equivalent to the value of the *st\_mtime* member of the *stat* structure for a  
 26811 file, as described in the **XSH** specification; the access time is equivalent to the value of *st\_atime*.

26812 The time used can be specified by the **-t** *time* option-argument, the corresponding time fields of  
 26813 the file referenced by the **-r** *ref\_file* option-argument, or the *date\_time* operand, as specified in  
 26814 the following sections. If none of these are specified, *touch* will use the current time (the value  
 26815 returned by the equivalent of the **XSH** specification *time()* function).

26816 For each *file* operand, *touch* will perform actions equivalent to the following functions defined in  
 26817 the **XSH** specification:

- 26818 1. If *file* does not exist, a *creat()* function call is made with the *file* operand used as the *path*  
 26819 argument and the value of the bitwise inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP,  
 26820 S\_IWGRP, S\_IROTH and S\_IWOTH used as the *mode* argument.
- 26821 2. The *utime()* function is called with the following arguments:
  - 26822 a. The *file* operand is used as the *path* argument.
  - 26823 b. The *utimbuf* structure members *actime* and *modtime* are determined as described in the  
 26824 OPTIONS section.

## 26825 OPTIONS

26826 The *touch* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

26827 The following options are supported:

- 26828 **-a** Change the access time of *file*. Do not change the modification time unless **-m** is also  
 26829 specified.
- 26830 **-c** Do not create a specified *file* if it does not exist. Do not write any diagnostic messages  
 26831 concerning this condition.
- 26832 **-m** Change the modification time of *file*. Do not change the access time unless **-a** is also  
 26833 specified.
- 26834 **-r** *ref\_file*  
 26835 Use the corresponding time of the file named by the pathname *ref\_file* instead of the  
 26836 current time.
- 26837 **-t** *time*  
 26838 Use the specified *time* instead of the current time. The option-argument will be a  
 26839 decimal number of the form:
- 26840 [ [ *CC*] *YY*] *MMDD* *hhmm* [ . *SS* ]
- 26841 where each two digits represents the following:
- 26842 *MM* The month of the year [01–12].
- 26843 *DD* The day of the month [01–31].

26844 *hh* The hour of the day [00–23].

26845 *mm* The minute of the hour [00–59].

26846 *CC* The first two digits of the year (the century).

26847 *YY* The second two digits of the year.

26848 *SS* The second of the minute [00–61].

26849 Both *CC* and *YY* are optional. If neither is given, the current year will be assumed. If  
26850 *YY* is specified, but *CC* is not, *CC* will be derived as follows:

26851

26852

26853

26854

| If <i>YY</i> is: | <i>CC</i> becomes: |
|------------------|--------------------|
| 69–99            | 19                 |
| 00–68            | 20                 |

26855 The resulting time will be affected by the value of the *TZ* environment variable. If the  
26856 resulting time value precedes the Epoch, *touch* will exit immediately with an error  
26857 status. The range of valid times past the Epoch is implementation-dependent, but will  
26858 extend to at least midnight 1 January 2000 UCT. Some systems will not be able to  
26859 represent dates beyond the January 18, 2038, because they use **signed int** as a time  
26860 holder.

26861 The range for *SS* is (00–61) rather than (00–59) because of leap seconds. If *SS* is 60 or 61,  
26862 and the resulting time, as affected by the *TZ* environment variable, does not refer to a  
26863 leap second, the resulting time will be one or two seconds after a time where *SS* is 59. If  
26864 *SS* is not given a value, it is assumed to be zero.

26865 If neither the **–a** nor **–m** options were specified, *touch* will behave as if both the **–a** and **–m**  
26866 options were specified.

## 26867 OPERANDS

26868 The following operands are supported:

26869 *file* A pathname of a file whose times are to be modified.

26870 OB *date\_time*

26871 Use the specified *date\_time* instead of the current time. The operand is a decimal  
26872 number of the form:

26873 *MMDDhhmm*[*yy*]

26874 where *MM*, *DD*, *hh*, and *mm* are as described for the *time* option-argument to the **–t**  
26875 option and the optional *yy* is interpreted as follows:

26876 If not specified, the current year will be used. If *yy* is in the range 69–99, the year  
26877 1969–1999, respectively, will be used. Otherwise, the results are unspecified.

26878 If no **–r** option is specified, no **–t** option is specified, at least two operands are specified,  
26879 and the first operand is an eight- or ten-digit decimal integer, the first operand will be  
26880 assumed to be a *date\_time* operand. Otherwise, the first operand will be assumed to be  
26881 a *file* operand.

## 26882 STDIN

26883 Not used.

## 26884 INPUT FILES

26885 None.

26886 **ENVIRONMENT VARIABLES**

26887 The following environment variables affect the execution of *touch*:

26888 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 26889 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 26890 default locale will be used. If any of the internationalisation variables contains an  
 26891 invalid setting, the utility will behave as if none of the variables had been defined.

26892 **LC\_ALL**

26893 If set to a non-empty string value, override the values of all the other  
 26894 internationalisation variables.

26895 **LC\_CTYPE**

26896 Determine the locale for the interpretation of sequences of bytes of text data as  
 26897 characters (for example, single- as opposed to multi-byte characters in arguments).

26898 **LC\_MESSAGES**

26899 Determine the locale that should be used to affect the format and contents of diagnostic  
 26900 messages written to standard error.

26901 EX **NLSPATH**

26902 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

26903 OB **TZ** Determine the timezone to be used for interpreting the *time* option-argument (or  
 26904 *date\_time* operand; see above).

26905 **ASYNCHRONOUS EVENTS**

26906 Default.

26907 **STDOUT**

26908 Not used.

26909 **STDERR**

26910 Used only for diagnostic messages.

26911 **OUTPUT FILES**

26912 None.

26913 **EXTENDED DESCRIPTION**

26914 None.

26915 **EXIT STATUS**

26916 The following exit values are returned:

26917 0 The utility executed successfully and all requested changes were made.

26918 >0 An error occurred.

26919 **CONSEQUENCES OF ERRORS**

26920 Default.

26921 **APPLICATION USAGE**

26922 The interpretation of time is taken to be **seconds since the Epoch** (see the **XBD** specification,  
 26923 **Chapter 2, Glossary**). It should be noted that implementations conforming to the **XSH**  
 26924 specification do not take leap seconds into account when computing seconds since the Epoch.  
 26925 When *SS=60* is used, the resulting time always refers to 1 plus “seconds since the Epoch” for a  
 26926 time when *SS=59*.

26927 Although the *-t time* option-argument and the obsolescent *date\_time* operand specify values in  
 26928 1969, the access time and modification time fields are defined in terms of seconds since the  
 26929 Epoch (midnight on 1 January 1970 UTC). Therefore, depending on the value of *TZ* when *touch*

26930 is run, there will never be more than a few valid hours in 1969 and there need not be any valid  
26931 times in 1969.

26932 One ambiguous situation occurs if `-t time` is not specified, `-r ref_file` is not specified, and the  
26933 first operand is an eight- or ten-digit decimal number. A portable script can avoid this problem  
26934 by using:

26935 `touch -- file`

26936 or:

26937 `touch ./file`

26938 in this case.

#### 26939 **EXAMPLES**

26940 None.

#### 26941 **FUTURE DIRECTIONS**

26942 The obsolescent `date_time` operand may be withdrawn in a future issue. Applications should use  
26943 the `-r` or `-t` options. |

#### 26944 **SEE ALSO**

26945 `date`, the XSH specification description of `creat()`, `time()`, `<sys/stat.h>`.

#### 26946 **CHANGE HISTORY**

26947 First released in Issue 2. |

#### 26948 **Issue 4**

26949 Aligned with the ISO/IEC 9945-2: 1993 standard.

26950 **NAME**

26951 tput — change terminal characteristics

26952 **SYNOPSIS**26953 tput [-T *type*] *operand*...26954 **DESCRIPTION**

26955 The *tput* utility displays terminal-dependent information. The manner in which this information  
 26956 is retrieved is unspecified. The information displayed will clear the terminal screen, initialise the  
 26957 user's terminal or reset the user's terminal, depending on the operand given. The exact  
 26958 consequences of displaying this information are unspecified.

26959 **OPTIONS**26960 The *tput* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

26961 The following option is supported:

26962 **-T *type*** Indicate the type of terminal. If this option is not supplied and the *TERM* variable is  
 26963 unset or null, an unspecified default terminal type will be used. The setting of *type* will  
 26964 take precedence over the value in *TERM*.

26965 **OPERANDS**

26966 The following strings will be supported as operands by the implementation in the POSIX locale:

26967 **clear** Display the clear-screen sequence.

26968 **init** Display the sequence that will initialise the user's terminal in an implementation-  
 26969 dependent manner.

26970 **reset** Display the sequence that will reset the user's terminal in an implementation-  
 26971 dependent manner.

26972 If a terminal does not support any of the operations described by these operands, this will not be  
 26973 considered an error condition.

26974 **STDIN**

26975 Not used.

26976 **INPUT FILES**

26977 None.

26978 **ENVIRONMENT VARIABLES**26979 The following environment variables affect the execution of *tput*:

26980 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 26981 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 26982 default locale will be used. If any of the internationalisation variables contains an  
 26983 invalid setting, the utility will behave as if none of the variables had been defined.

26984 **LC\_ALL**

26985 If set to a non-empty string value, override the values of all the other  
 26986 internationalisation variables.

26987 **LC\_CTYPE**

26988 Determine the locale for the interpretation of sequences of bytes of text data as  
 26989 characters (for example, single- as opposed to multi-byte characters in arguments).

26990 **LC\_MESSAGES**

26991 Determine the locale that should be used to affect the format and contents of diagnostic  
 26992 messages written to standard error.

26993 EX **NLSPATH**  
 26994 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

26995 **TERM** Determine the terminal type. If this variable is unset or null, and if the **-T** option is not  
 26996 specified, an unspecified default terminal type will be used.

26997 **ASYNCHRONOUS EVENTS**  
 26998 Default.

26999 **STDOUT**  
 27000 If standard output is a terminal device, it may be used for writing the appropriate sequence to  
 27001 clear the screen or reset or initialise the terminal. If standard output is not a terminal device,  
 27002 undefined results occur.

27003 **STDERR**  
 27004 Used only for diagnostic messages.

27005 **OUTPUT FILES**  
 27006 None.

27007 **EXTENDED DESCRIPTION**  
 27008 None.

27009 **EXIT STATUS**  
 27010 The following exit values are returned:  
 27011 0 The requested string was written successfully.  
 27012 1 Unspecified.  
 27013 2 Usage error.  
 27014 3 No information is available about the specified terminal type.  
 27015 4 The specified operand is invalid.  
 27016 >4 An error occurred.

27017 **CONSEQUENCES OF ERRORS**  
 27018 If one of the operands is not available for the terminal, *tput* continues processing the remaining  
 27019 operands.

27020 **APPLICATION USAGE**  
 27021 The difference between resetting and initialising a terminal is left unspecified, as they vary  
 27022 greatly based on hardware types. In general, resetting is a more severe action.  
 27023 Some terminals use control characters to perform the stated functions, and on such terminals it  
 27024 might make sense to use *tput* to store the initialisation strings in a file or environment variable  
 27025 for later use. However, because other terminals might rely on system calls to do this work, the  
 27026 standard output cannot be used in a portable manner, such as the following non-portable  
 27027 constructs:  
 27028 ClearVar='tput clear'  
 27029 tput reset | mailx -s "Wake Up" ddg

27030 **EXAMPLES**  
 27031 1. Initialise the terminal according to the type of terminal in the environmental variable  
 27032 **TERM**. This command can be included in a **.profile** file.  
 27033 tput init  
 27034 2. Reset a 450 terminal.  
 27035 tput -T 450 reset

27036 **FUTURE DIRECTIONS**

27037       None.

27038 **SEE ALSO**27039       *stty, tabs.*27040 **CHANGE HISTORY**

27041       First released in Issue 4.



27042 **NAME**

27043       tr — translate characters

27044 **SYNOPSIS**27045       tr [-cs] *string1 string2*27046       tr -s[-c] *string1*27047       tr -d[-c] *string1*27048       tr -ds[-c] *string1 string2*27049 **DESCRIPTION**

27050       The *tr* utility copies the standard input to the standard output with substitution or deletion of  
 27051       selected characters. The options specified and the *string1* and *string2* operands control  
 27052       translations that occur while copying characters and single-character collating elements.

27053 **OPTIONS**27054       The *tr* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

27055       The following options are supported:

27056       -c       Complement the set of characters specified by *string1*. See the EXTENDED  
 27057       DESCRIPTION section.

27058       -d       Delete all occurrences of input characters that are specified by *string1*.

27059       -s       Replace instances of repeated characters with a single character, as described in the  
 27060       EXTENDED DESCRIPTION section.

27061 **OPERANDS**

27062       The following operands are supported:

27063       *string1*

27064       *string2*   Translation control strings. Each string represents a set of characters to be converted  
 27065       into an array of characters used for the translation. For a detailed description of how  
 27066       the strings are interpreted, see the EXTENDED DESCRIPTION section.

27067 **STDIN**

27068       The standard input can be any type of file.

27069 **INPUT FILES**

27070       None.

27071 **ENVIRONMENT VARIABLES**27072       The following environment variables affect the execution of *tr*:

27073       *LANG*    Provide a default value for the internationalisation variables that are unset or null. If  
 27074       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 27075       default locale will be used. If any of the internationalisation variables contains an  
 27076       invalid setting, the utility will behave as if none of the variables had been defined.

27077       *LC\_ALL*

27078       If set to a non-empty string value, override the values of all the other  
 27079       internationalisation variables.

27080       *LC\_COLLATE*

27081       Determine the locale for the behaviour of range expressions and equivalence classes.

|          |                             |                                                                                                                  |
|----------|-----------------------------|------------------------------------------------------------------------------------------------------------------|
| 27082    | <b>LC_CTYPE</b>             |                                                                                                                  |
| 27083    |                             | Determine the locale for the interpretation of sequences of bytes of text data as                                |
| 27084    |                             | characters (for example, single- versus multi-byte characters in arguments) and the                              |
| 27085    |                             | behaviour of character classes.                                                                                  |
| 27086    | <b>LC_MESSAGES</b>          |                                                                                                                  |
| 27087    |                             | Determine the locale that should be used to affect the format and contents of diagnostic                         |
| 27088    |                             | messages written to standard error.                                                                              |
| 27089 EX | <b>NLSPATH</b>              |                                                                                                                  |
| 27090    |                             | Determine the location of message catalogues for the processing of <b>LC_MESSAGES</b> .                          |
| 27091    | <b>ASYNCHRONOUS EVENTS</b>  |                                                                                                                  |
| 27092    |                             | Default.                                                                                                         |
| 27093    | <b>STDOUT</b>               |                                                                                                                  |
| 27094    |                             | The <i>tr</i> output is identical to the input, with the exception of the specified transformations.             |
| 27095    | <b>STDERR</b>               |                                                                                                                  |
| 27096    |                             | Used only for diagnostic messages.                                                                               |
| 27097    | <b>OUTPUT FILES</b>         |                                                                                                                  |
| 27098    |                             | None.                                                                                                            |
| 27099    | <b>EXTENDED DESCRIPTION</b> |                                                                                                                  |
| 27100    |                             | The operands <i>string1</i> and <i>string2</i> (if specified) define two arrays of characters. The constructs in |
| 27101    |                             | the following list can be used to specify characters or single-character collating elements. If any              |
| 27102    |                             | of the constructs result in multi-character collating elements, <i>tr</i> will exclude, without a                |
| 27103    |                             | diagnostic, those multi-character elements from the resulting array.                                             |
| 27104    | <i>character</i>            |                                                                                                                  |
| 27105    |                             | Any character not described by one of the conventions below represents itself.                                   |
| 27106    | <i>\octal</i>               | Octal sequences can be used to represent characters with specific coded values. An                               |
| 27107    |                             | octal sequence consists of a backslash followed by the longest sequence of one-, two- or                         |
| 27108    |                             | three-octal-digit characters (01234567). The sequence causes the character whose                                 |
| 27109    |                             | encoding is represented by the one-, two- or three-digit octal integer to be placed into                         |
| 27110    |                             | the array. If the size of a byte on the system is greater than nine bits, the valid escape                       |
| 27111    |                             | sequence used to represent a byte is implementation-dependent. Multi-byte characters                             |
| 27112    |                             | require multiple, concatenated escape sequences of this type, including the leading \                            |
| 27113    |                             | for each byte.                                                                                                   |
| 27114    | <i>\character</i>           |                                                                                                                  |
| 27115    |                             | The backslash-escape sequences in the table in the <b>XBD</b> specification, <b>Chapter 3, File</b>              |
| 27116    |                             | <b>Format Notation</b> (\, \a, \b, \f, \n, \r, \t, \v) are supported. The results of using any                   |
| 27117    |                             | other character, other than an octal digit, following the backslash are unspecified.                             |
| 27118    | <i>c-c</i>                  | Represents the range of collating elements between the range endpoints, inclusive, as                            |
| 27119    |                             | defined by the current setting of the <b>LC_COLLATE</b> locale category. The starting                            |
| 27120    |                             | endpoint must precede the second endpoint in the current collation order. The                                    |
| 27121    |                             | characters or collating elements in the range are placed in the array in ascending                               |
| 27122    |                             | collation sequence.                                                                                              |

27123 [:class:] Represents all characters belonging to the defined character class, as defined by the  
 27124 current setting of the LC\_CTYPE locale category. The following character class names  
 27125 will be accepted when specified in *string1*:

|       |              |              |              |              |              |               |
|-------|--------------|--------------|--------------|--------------|--------------|---------------|
| 27126 | <b>alnum</b> | <b>blank</b> | <b>digit</b> | <b>lower</b> | <b>punct</b> | <b>upper</b>  |
| 27127 | <b>alpha</b> | <b>cntrl</b> | <b>graph</b> | <b>print</b> | <b>space</b> | <b>xdigit</b> |

27128 EX In addition, character class expressions of the form [ :*name*:] are recognised in those  
 27129 locales where the *name* keyword has been given a **charclass** definition in the  
 27130 LC\_CTYPE category.

27131 When both the **-d** and **-s** options are specified, any of the character class names will be  
 27132 accepted in *string2*. Otherwise, only character class names **lower** or **upper** are valid in  
 27133 *string2* and then only if the corresponding character class (upper and **lower**,  
 27134 respectively) is specified in the same relative position in *string1*. Such a specification is  
 27135 interpreted as a request for case conversion. When [:lower:] appears in *string1* and  
 27136 [:upper:] appears in *string2*, the arrays will contain the characters from the **toupper**  
 27137 mapping in the LC\_CTYPE category of the current locale. When [:upper:] appears in  
 27138 *string1* and [:lower:] appears in *string2*, the arrays will contain the characters from the  
 27139 **tolower** mapping in the LC\_CTYPE category of the current locale. The first character  
 27140 from each mapping pair will be in the array for *string1* and the second character from  
 27141 each mapping pair will be in the array for *string2* in the same relative position.

27142 Except for case conversion, the characters specified by a character class expression are  
 27143 placed in the array in an unspecified order.

27144 If the name specified for *class* does not define a valid character class in the current  
 27145 locale, the behaviour is undefined.

27146 [=equiv=]  
 27147 Represents all characters or collating elements belonging to the same equivalence class  
 27148 as *equiv*, as defined by the current setting of the LC\_COLLATE locale category. An  
 27149 equivalence class expression is allowed only in *string1*, or in *string2* when it is being  
 27150 used by the combined **-d** and **-s** options. The characters belonging to the equivalence  
 27151 class are placed in the array in an unspecified order.

27152 [*x*\**n*] Represents *n* repeated occurrences of the character *x*. Because this expression is used  
 27153 to map multiple characters to one, it is only valid when it occurs in *string2*. If *n* is  
 27154 omitted or is zero, it is interpreted as large enough to extend the *string2*-based sequence  
 27155 to the length of the *string1*-based sequence. If *n* has a leading zero, it is interpreted as  
 27156 an octal value. Otherwise, it is interpreted as a decimal value.

27157 When the **-d** option is not specified:

- 27158 • Each input character found in the array specified by *string1* is replaced by the character in the  
 27159 same relative position in the array specified by *string2*. When the array specified by *string2* is  
 27160 shorter than the one specified by *string1*, the results are unspecified.
- 27161 • If the **-c** option is specified, the complements of the characters specified by *string1* (the set of  
 27162 all characters in the current character set, as defined by the current setting of LC\_CTYPE,  
 27163 except for those actually specified in the *string1* operand) are placed in the array in ascending  
 27164 collation sequence, as defined by the current setting of LC\_COLLATE.
- 27165 • Because the order in which characters specified by character class expressions or equivalence  
 27166 class expressions is undefined, such expressions should only be used if the intent is to map  
 27167 several characters into one. An exception is case conversion, as described previously.

27168 When the **-d** option is specified:

- 27169 • Input characters found in the array specified by *string1* will be deleted.
- 27170 • When the **-c** option is specified with **-d**, all characters except those specified by *string1* will
- 27171 be deleted. The contents of *string2* will be ignored, unless the **-s** option is also specified.
- 27172 • The same string cannot be used for both the **-d** and the **-s** option; when both options are
- 27173 specified, both *string1* (used for deletion) and *string2* (used for squeezing) are required.

27174 When the **-s** option is specified, after any deletions or translations have taken place, repeated  
 27175 sequences of the same character will be replaced by one occurrence of the same character, if the  
 27176 character is found in the array specified by the last operand. If the last operand contains a  
 27177 character class, such as the following example:

```
27178 tr -s '[:space:]'
```

27179 the last operand's array will contain all of the characters in that character class. However, in a  
 27180 case conversion, as described previously, such as:

```
27181 tr -s '[:upper:]' '[:lower:]'
```

27182 the last operand's array will contain only those characters defined as the second characters in  
 27183 each of the **toupper** or **tolower** character pairs, as appropriate.

27184 An empty string used for *string1* or *string2* produces undefined results.

#### 27185 EXIT STATUS

27186 The following exit values are returned:

- 27187 0 All input was processed successfully.
- 27188 >0 An error occurred.

#### 27189 CONSEQUENCES OF ERRORS

27190 Default.

#### 27191 APPLICATION USAGE

27192 If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

27193 If an ordinary digit (representing itself) is to follow an octal sequence, the octal sequence must  
 27194 use the full three digits to avoid ambiguity.

27195 When *string2* is shorter than *string1*, a difference results between historical System V and BSD  
 27196 systems. A BSD system will pad *string2* with the last character found in *string2*. Thus, it is  
 27197 possible to do the following:

```
27198 tr 0123456789 d
```

27199 which would translate all digits to the letter d. Since this area is specifically unspecified in the  
 27200 document, both the BSD and System V behaviours are allowed, but a portable application  
 27201 cannot rely on the BSD behaviour. It would have to code the example in the following way:

```
27202 tr 0123456789 '[d*]'
```

27203 It should be noted that, despite similarities in appearance, the string operands used by *tr* are not  
 27204 regular expressions.

27205 Unlike some previous versions, the Issue 4 *tr* correctly processes NUL characters in its input  
 27206 stream. NUL characters can be stripped by using:

```
27207 tr -d '\000'
```

27208 **EXAMPLES**

- 27209           1. The following example creates a list of all words in *file1* one per line in *file2*, where a word  
 27210 is taken to be a maximal string of letters.

27211           tr -cs "[:alpha:]" "[\n\*]" <file1 >file2

- 27212           2. The next example translates all lower-case characters in **file1** to upper-case and writes the  
 27213 results to standard output.

27214           tr "[:lower:]" "[:upper:]" <file1

27215           Note that the caveat expressed in the corresponding Issue 3 example is no longer in effect.  
 27216 This case conversion is now a special case that employs the **tolower** and **toupper**  
 27217 classifications, ensuring that proper mapping is accomplished (when the locale is correctly  
 27218 defined).

- 27219           3. This example uses an equivalence class to identify accented variants of the base character e  
 27220 in **file1**, which are stripped of diacritical marks and written to **file2**.

27221           tr "[=e=]" e <file1 >file2

27222 **FUTURE DIRECTIONS**

27223           None.

27224 **SEE ALSO**

27225           *sed*.

27226 **CHANGE HISTORY**

27227           First released in Issue 2.

27228 **Issue 4**

27229           Aligned with the ISO/IEC 9945-2: 1993 standard.

27230 **NAME**

27231 true — return true value

27232 **SYNOPSIS**

27233 true

27234 **DESCRIPTION**27235 The *true* utility will return with exit code zero.27236 **OPTIONS**

27237 None.

27238 **OPERANDS**

27239 None.

27240 **STDIN**

27241 Not used.

27242 **INPUT FILES**

27243 None.

27244 **ENVIRONMENT VARIABLES**

27245 None.

27246 **ASYNCHRONOUS EVENTS**

27247 Default.

27248 **STDOUT**

27249 Not used.

27250 **STDERR**

27251 None.

27252 **OUTPUT FILES**

27253 None.

27254 **EXTENDED DESCRIPTION**

27255 None.

27256 **EXIT STATUS**

27257 Default.

27258 **CONSEQUENCES OF ERRORS**

27259 None.

27260 **APPLICATION USAGE**

27261 This utility is typically used in shell scripts, as shown in the the EXAMPLES section. The special  
27262 built-in utility ":" is sometimes more efficient than *true*.

27263 **EXAMPLES**

27264 This command will be executed forever:

```
27265 while true
27266 do
27267 command
27268 done
```

27269 **FUTURE DIRECTIONS**

27270 None.

27271 **SEE ALSO**27272 *false*, Section 2.9 on page 45.27273 **CHANGE HISTORY**

27274 First released in Issue 2.

27275 **Issue 4**

27276 Aligned with the ISO/IEC 9945-2: 1993 standard.

27277 **NAME**

27278           tsort — topological sort

27279 **SYNOPSIS**27280 EX       tsort [*file*]27281 **DESCRIPTION**27282           The *tsort* utility writes to standard output a totally ordered list of items consistent with a partial  
27283           ordering of items contained in the input.27284           The input consists of pairs of items (non-empty strings) separated by blanks. Pairs of different  
27285           items indicate ordering. Pairs of identical items indicate presence, but not ordering.27286 **OPTIONS**

27287           None.

27288 **OPERANDS**

27289           The following operand is supported:

27290           *file*       A pathname of a text file to order. If no *file* operand is given, the standard input is  
27291           used.27292 **STDIN**27293           The standard input is a text file that is used if no *file* operand is given.27294 **INPUT FILES**27295           The input file named by the *file* operand is a text file.27296 **ENVIRONMENT VARIABLES**27297           The following environment variables affect the execution of *tsort*:27298           *LANG*       Provide a default value for the internationalisation variables that are unset or null. If  
27299           *LANG* is unset or null, the corresponding value from the implementation-dependent  
27300           default locale will be used. If any of the internationalisation variables contains an  
27301           invalid setting, the utility will behave as if none of the variables had been defined.27302           *LC\_ALL*27303                       If set to a non-empty string value, override the values of all the other  
27304           internationalisation variables.27305           *LC\_CTYPE*27306                       Determine the locale for the interpretation of sequences of bytes of text data as  
27307           characters (for example, single- as opposed to multi-byte characters in arguments and  
27308           input files).27309           *LC\_MESSAGES*27310                       Determine the locale that should be used to affect the format and contents of diagnostic  
27311           messages written to standard error.27312           *NLSPATH*27313                       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.27314 **ASYNCHRONOUS EVENTS**

27315           Default.

27316 **STDOUT**27317           The standard output is a text file consisting of the order list produced from the partially ordered  
27318           input.



27319 **STDERR**

27320           Used only for diagnostic messages.

27321 **OUTPUT FILES**

27322           None.

27323 **EXTENDED DESCRIPTION**

27324           None.

27325 **EXIT STATUS**

27326           The following exit values are returned:

27327           0   Successful completion.

27328           &gt;0  An error occurred.

27329 **CONSEQUENCES OF ERRORS**

27330           Default.

27331 **APPLICATION USAGE**

27332           The *LC\_COLLATE* variable need not affect the actions of *tsort*. The output ordering is not  
 27333           lexicographic, but depends on the pairs of items given as input.

27334 **EXAMPLES**

27335           The command:

```
27336 tsort <<EOF
27337 a b c c d e
27338 g g
27339 f g e f
27340 h h
27341 EOF
```

27342           produces the output:

```
27343 a
27344 b
27345 c
27346 d
27347 e
27348 f
27349 g
27350 h
```

27351 **FUTURE DIRECTIONS**

27352           None.

27353 **SEE ALSO**

27354           None.

27355 **CHANGE HISTORY**

27356           First released in Issue 2.

27357 **Issue 4**

27358           Format reorganised.

27359           Internationalised environment variable support mandated.

27360 **NAME**

27361           tty — return user's terminal name

27362 **SYNOPSIS**

27363           tty

27364 OB       tty -s

27365

27366 **DESCRIPTION**

27367           The *tty* utility writes to the standard output the name of the terminal that is open as standard  
 27368           input. The name that is used is equivalent to the string that would be returned by the **XSH**  
 27369           specification *ttyname()* function.

27370 **OPTIONS**27371           The *tty* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

27372           The following option is supported:

|          |           |                                                                                                                                                                                                           |
|----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27373 OB | <b>-s</b> | Do not write the terminal name. Only the exit status will be affected by this option.<br>The terminal status will be determined as if the <b>XSH</b> specification <i>isatty()</i> function<br>were used. |
| 27374    |           |                                                                                                                                                                                                           |
| 27375    |           |                                                                                                                                                                                                           |

27376 **OPERANDS**

27377           None.

27378 **STDIN**

27379           While no input is read from standard input, standard input will be examined to determine  
 27380           whether or not it is a terminal, and, if so, to determine the name of the terminal.

27381 **INPUT FILES**

27382           None.

27383 **ENVIRONMENT VARIABLES**27384           The following environment variables affect the execution of *tty*:

|                                  |             |                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27385<br>27386<br>27387<br>27388 | <b>LANG</b> | Provide a default value for the internationalisation variables that are unset or null. If<br><i>LANG</i> is unset or null, the corresponding value from the implementation-dependent<br>default locale will be used. If any of the internationalisation variables contains an<br>invalid setting, the utility will behave as if none of the variables had been defined. |
|----------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

27389           **LC\_ALL**

|                |  |                                                                                                             |
|----------------|--|-------------------------------------------------------------------------------------------------------------|
| 27390<br>27391 |  | If set to a non-empty string value, override the values of all the other<br>internationalisation variables. |
|----------------|--|-------------------------------------------------------------------------------------------------------------|

27392           **LC\_CTYPE**

|                |  |                                                                                                                                                                          |
|----------------|--|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27393<br>27394 |  | Determine the locale for the interpretation of sequences of bytes of text data as<br>characters (for example, single- as opposed to multi-byte characters in arguments). |
|----------------|--|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

27395           **LC\_MESSAGES**

|                         |  |                                                                                                                                                                                        |
|-------------------------|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27396<br>27397<br>27398 |  | Determine the locale that should be used to affect the format and contents of diagnostic<br>messages written to standard error and informative messages written to standard<br>output. |
|-------------------------|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

27399 EX       **NLSPATH**27400           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.27401 **ASYNCHRONOUS EVENTS**

27402           Default.

27403 **STDOUT**

27404 **OB** If the **-s** option is specified, standard output will not be used. If the **-s** option is not specified and  
 27405 standard input is a terminal device, a pathname of the terminal as specified by the **XSH**  
 27406 specification *ttname()* will be written in the following format:

27407       "*%s*\n", *<terminal name>*

27408       Otherwise, a message will be written indicating that standard input is not connected to a  
 27409 terminal. In the POSIX locale, the *tty* utility will use the format:

27410       "not a tty\n"

27411 **STDERR**

27412       Used only for diagnostic messages.

27413 **OUTPUT FILES**

27414       None.

27415 **EXTENDED DESCRIPTION**

27416       None.

27417 **EXIT STATUS**

27418       The following exit values are returned:

27419       0   Standard input is a terminal.

27420       1   Standard input is not a terminal.

27421       >1   An error occurred.

27422 **CONSEQUENCES OF ERRORS**

27423       Default.

27424 **APPLICATION USAGE**

27425       This utility checks the status of the file open as standard input against that of a system-defined  
 27426 set of files. It is possible that no match can be found, or that the match found need not be the  
 27427 same file as that which was opened for standard input (although they are the same device).

27428       The **-s** option is useful only if the exit code is wanted. It does not rely on the ability to form a  
 27429 valid pathname. Portable applications should use *test -t 0*.

27430 **EXAMPLES**

27431       None.

27432 **FUTURE DIRECTIONS**

27433       None.

27434 **SEE ALSO**

27435       The **XSH** specification description of *isatty()*, *ttname()*.

27436 **CHANGE HISTORY**

27437       First released in Issue 2.

27438 **Issue 4**

27439       Aligned with the ISO/IEC 9945-2: 1993 standard.

27440 **Issue 5**

27441       The SYNOPSIS is changed to indicate two forms of the command, with the second form marked  
 27442 as obsolete. This is a clarification and does not change the functionality published in previous  
 27443 issues.

27444 **NAME**

27445           type — write a description of command type

27446 **SYNOPSIS**

27447 EX       type name...

27448 **DESCRIPTION**

27449           The *type* utility indicates how each argument would be interpreted if used as a command name.

27450 **OPTIONS**

27451           None.

27452 **OPERANDS**

27453           The following operand is supported:

27454       *name*     A name to be interpreted.

27455 **STDIN**

27456           Not used.

27457 **INPUT FILES**

27458           None.

27459 **ENVIRONMENT VARIABLES**

27460           The following environment variables affect the execution of *type*:

27461       *LANG*     Provide a default value for the internationalisation variables that are unset or null. If  
27462                   *LANG* is unset or null, the corresponding value from the implementation-dependent  
27463                   default locale will be used. If any of the internationalisation variables contains an  
27464                   invalid setting, the utility will behave as if none of the variables had been defined.

27465       *LC\_ALL*

27466                   If set to a non-empty string value, override the values of all the other  
27467                   internationalisation variables.

27468       *LC\_CTYPE*

27469                   Determine the locale for the interpretation of sequences of bytes of text data as  
27470                   characters (for example, single- as opposed to multi-byte characters in arguments).

27471       *LC\_MESSAGES*

27472                   Determine the locale that should be used to affect the format and contents of diagnostic  
27473                   messages written to standard error.

27474       *NLSPATH*

27475                   Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

27476       *PATH*     Determine the location of *name*, as described in the **XBD** specification, **Chapter 6**,  
27477                   **Environment Variables**.

27478 **ASYNCHRONOUS EVENTS**

27479           Default.

27480 **STDOUT**

27481           The standard output of *type* contains information about each operand in an unspecified format.

27482           The information provided typically identifies the operand as a shell built-in, function, alias or  
27483           keyword, and where applicable, may display the operand's pathname.

27484 **STDERR**

27485           Used only for diagnostic messages.

27486 **OUTPUT FILES**

27487       None.

27488 **EXTENDED DESCRIPTION**

27489       None.

27490 **EXIT STATUS**

27491       The following exit values are returned:

27492       0   Successful completion.

27493       &gt;0  An error occurred.

27494 **CONSEQUENCES OF ERRORS**

27495       Default.

27496 **APPLICATION USAGE**

27497       Since *type* must be aware of the contents of the current shell execution environment (such as the  
27498       lists of commands, functions and built-ins processed by *hash*), it is always provided as a shell  
27499       regular built-in. If it is called in a separate utility execution environment, such as one of the  
27500       following:

27501       nohup type writer

27502       find . -type f | xargs type

27503       it might not produce accurate results.

27504 **EXAMPLES**

27505       None.

27506 **FUTURE DIRECTIONS**

27507       None.

27508 **SEE ALSO**27509       *command*.27510 **CHANGE HISTORY**

27511       First released in Issue 2.

27512 **Issue 4**27513       Relocated from the *sh* description to reflect its status as a regular built-in utility.

27514 **NAME**

27515           ulimit — set or report file size limit

27516 **SYNOPSIS**27517 EX       ulimit [-f][*blocks*]27518 **DESCRIPTION**

27519       The *ulimit* utility sets or reports the file-size writing limit imposed on files written by the shell  
 27520       and its child processes (files of any size may be read). Only a process with appropriate  
 27521       privileges can increase the limit.

27522 **OPTIONS**27523       The *ulimit* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

27524       The following option is supported:

27525       -f       Set (or report, if no *blocks* operand is present), the file size limit in blocks. The -f option  
 27526       is also the default case.

27527 **OPERANDS**

27528       The following operand is supported:

27529       *blocks*   The number of 512-byte blocks to use as the new file size limit.27530 **STDIN**

27531       Not used.

27532 **INPUT FILES**

27533       None.

27534 **ENVIRONMENT VARIABLES**27535       The following environment variables affect the execution of *ulimit*:

27536       *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 27537       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 27538       default locale will be used. If any of the internationalisation variables contains an  
 27539       invalid setting, the utility will behave as if none of the variables had been defined.

27540       *LC\_ALL*

27541       If set to a non-empty string value, override the values of all the other  
 27542       internationalisation variables.

27543       *LC\_CTYPE*

27544       Determine the locale for the interpretation of sequences of bytes of text data as  
 27545       characters (for example, single- as opposed to multi-byte characters in arguments).

27546       *LC\_MESSAGES*

27547       Determine the locale that should be used to affect the format and contents of diagnostic  
 27548       messages written to standard error.

27549       *NLSPATH*27550       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.27551 **ASYNCHRONOUS EVENTS**

27552       Default.

27553 **STDOUT**

27554           The standard output is used when no *blocks* operand is present. If the current number of blocks  
27555           is limited, the number of blocks in the current limit is written in the following format:

27556           "%d\n", <number of 512-byte blocks>

27557           If there is no current limit on the number of blocks, in the POSIX locale the following format is  
27558           used:

27559           "unlimited\n"

27560 **STDERR**

27561           Used only for diagnostic messages.

27562 **OUTPUT FILES**

27563           None.

27564 **EXTENDED DESCRIPTION**

27565           None.

27566 **EXIT STATUS**

27567           The following exit values are returned:

27568           0   Successful completion.

27569           >0  A request for a higher limit was rejected or an error occurred.

27570 **CONSEQUENCES OF ERRORS**

27571           Default.

27572 **APPLICATION USAGE**

27573           Since *ulimit* affects the current shell execution environment, it is always provided as a shell  
27574           regular built-in. If it is called in separate utility execution environment, such as one of the  
27575           following:

27576           nohup ulimit -f 10000

27577           env ulimit 10000

27578           it will not affect the file size limit of the caller's environment.

27579           Once a limit has been decreased by a process, it cannot be increased (unless appropriate  
27580           privileges are involved), even back to the original system limit.

27581 **EXAMPLES**

27582           Set the file size limit to 51,200 bytes:

27583           ulimit -f 100

27584 **FUTURE DIRECTIONS**

27585           None.

27586 **SEE ALSO**

27587           The **XSH** specification description of *ulimit()*.

27588 **CHANGE HISTORY**

27589           First released in Issue 2.

27590 **Issue 4**

27591           Relocated from the *sh* description to reflect its status as a regular built-in utility.

27592 **NAME**

27593           umask — get or set the file mode creation mask

27594 **SYNOPSIS**27595           umask [-S][*mask*]27596 **DESCRIPTION**

27597           The *umask* utility will set the file mode creation mask of the current shell execution environment (see Section 2.12 on page 63) to the value specified by the *mask* operand. This mask will affect the initial value of the file permission bits of subsequently created files. If *umask* is called in a subshell or separate utility execution environment, such as one of the following:

```
27601 (umask 002)
27602 nohup umask ...
27603 find . -exec umask ... \;
```

27604           it will not affect the file mode creation mask of the caller's environment.

27605           If the *mask* operand is not specified, the *umask* utility will write to standard output the value of the invoking process's file mode creation mask.

27607 **OPTIONS**27608           The *umask* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

27609           The following option is supported:

27610           **-S**       Produce symbolic output.

27611           The default output style is unspecified, but will be recognised on a subsequent invocation of *umask* on the same system as a *mask* operand to restore the previous file mode creation mask.

27613 **OPERANDS**

27614           The following operand is supported:

27615           ***mask***     A string specifying the new file mode creation mask. The string is treated in the same way as the *mode* operand described in the the EXTENDED DESCRIPTION section for *chmod*.

27618           For a *symbolic\_mode* value, the new value of the file mode creation mask will be the logical complement of the file permission bits portion of the file mode specified by the *symbolic\_mode* string.

27621           In a *symbolic\_mode* value, the permissions *op* characters "+" and "-" will be interpreted relative to the current file mode creation mask; "+" will cause the bits for the indicated permissions to be cleared in the mask; "-" will cause the bits for the indicated permissions to be set in the mask.

27625           The interpretation of *mode* values that specify file mode bits other than the file permission bits is unspecified.

27627 EX           In the obsolescent octal integer form of *mode*, the specified bits will be set in the file mode creation mask.

27629           The file mode creation mask will be set to the resulting numeric value.

27630           The default output of a prior invocation of *umask* on the same system with no operand will also be recognised as a *mask* operand. The use of an operand obtained in this way is not obsolescent, even if it is an octal number.

27633 **STDIN**

27634           Not used.



27635 **INPUT FILES**

27636       None.

27637 **ENVIRONMENT VARIABLES**27638       The following environment variables affect the execution of *umask*:

27639       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 27640       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 27641       default locale will be used. If any of the internationalisation variables contains an  
 27642       invalid setting, the utility will behave as if none of the variables had been defined.

27643       **LC\_ALL**

27644       If set to a non-empty string value, override the values of all the other  
 27645       internationalisation variables.

27646       **LC\_CTYPE**

27647       Determine the locale for the interpretation of sequences of bytes of text data as  
 27648       characters (for example, single- as opposed to multi-byte characters in arguments).

27649       **LC\_MESSAGES**

27650       Determine the locale that should be used to affect the format and contents of diagnostic  
 27651       messages written to standard error.

27652 EX      **NLSPATH**27653       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.27654 **ASYNCHRONOUS EVENTS**

27655       Default.

27656 **STDOUT**

27657       When the *mask* operand is not specified, the *umask* utility will write a message to standard  
 27658       output that can later be used as a *umask mask* operand.

27659       If **-S** is specified, the message will be in the following format:

27660       "u=%s,g=%s,o=%s\n", <owner permissions>, <group permissions>,  
 27661       <other permissions>

27662       where the three values will be combinations of letters from the set {r, w, x}; the presence of a  
 27663       letter will indicate that the corresponding bit is clear in the file mode creation mask.

27664       If a *mask* operand is specified, there will be no output written to standard output.27665 **STDERR**

27666       Used only for diagnostic messages.

27667 **OUTPUT FILES**

27668       None.

27669 **EXTENDED DESCRIPTION**

27670       None.

27671 **EXIT STATUS**

27672       The following exit values are returned:

27673       0   The file mode creation mask was successfully changed, or no *mask* operand was supplied.  
 27674       >0   An error occurred.

27675 **CONSEQUENCES OF ERRORS**

27676       Default.

27677 **APPLICATION USAGE**

27678 Since *umask* affects the current shell execution environment, it is generally provided as a shell  
 27679 regular built-in.

27680 In contrast to the negative permission logic provided by the file mode creation mask and the  
 27681 octal number form of the *mask* argument, the symbolic form of the *mask* argument specifies those  
 27682 permissions that are left alone.

27683 The references to octal modes are marked EX because, although they are obsolescent in the  
 27684 ISO/IEC 9945-2:1993 standard, XSI-conformant systems have committed to maintaining them  
 27685 for portable applications until further notice.

27686 **EXAMPLES**

27687 Either of the commands:

27688 `umask a=rx,ug+w`

27689 `umask 002`

27690 sets the mode mask so that subsequently created files have their S\_IWOTH bit cleared.

27691 After setting the mode mask with either of the above commands, the *umask* command can be  
 27692 used to write out the current value of the mode mask:

27693 `$ umask`

27694 `0002`

27695 (The output format is unspecified, but historical implementations use the obsolescent octal  
 27696 integer mode format.)

27697 `$ umask -S`

27698 `u=rwx,g=rwx,o=rx`

27699 Either of these outputs can be used as the mask operand to a subsequent invocation of the *umask*  
 27700 utility.

27701 Assuming the mode mask is set as above, the command:

27702 `umask g-w`

27703 sets the mode mask so that subsequently created files have their S\_IWGRP, and S\_IWOTH bits  
 27704 cleared.

27705 The command:

27706 `umask -- -w`

27707 sets the mode mask so that subsequently created files have all their write bits cleared. Note that  
 27708 *mask* operands `-r`, `-w`, `-x` or anything beginning with a hyphen, must be preceded by `--` to keep  
 27709 it from being interpreted as an option.

27710 **FUTURE DIRECTIONS**

27711 None.

27712 **SEE ALSO**

27713 *chmod*, the XSH specification description of *umask*().

27714 **CHANGE HISTORY**

27715 First released in Issue 2.

27716 **Issue 4**

27717 Aligned with the ISO/IEC 9945-2: 1993 standard.

27718 **NAME**

27719 unalias — remove alias definitions

27720 **SYNOPSIS**27721 unalias *alias-name*...

27722 unalias -a

27723 **DESCRIPTION**

27724 The *unalias* utility removes the definition for each alias name specified. See Section 2.3.1 on page  
 27725 24. The aliases are removed from the current shell execution environment; see Section 2.12 on  
 27726 page 63.

27727 **OPTIONS**27728 The *unalias* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

27729 The following option is supported:

27730 **-a** Remove all alias definitions from the current shell execution environment.27731 **OPERANDS**

27732 The following operand is supported:

27733 *alias-name*

27734 The name of an alias to be removed.

27735 **STDIN**

27736 Not used.

27737 **INPUT FILES**

27738 None.

27739 **ENVIRONMENT VARIABLES**27740 The following environment variables affect the execution of *unalias*:

27741 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 27742 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 27743 default locale will be used. If any of the internationalisation variables contains an  
 27744 invalid setting, the utility will behave as if none of the variables had been defined.

27745 **LC\_ALL**

27746 If set to a non-empty string value, override the values of all the other  
 27747 internationalisation variables.

27748 **LC\_CTYPE**

27749 Determine the locale for the interpretation of sequences of bytes of text data as  
 27750 characters (for example, single- as opposed to multi-byte characters in arguments).

27751 **LC\_MESSAGES**

27752 Determine the locale that should be used to affect the format and contents of diagnostic  
 27753 messages written to standard error.

27754 EX **NLSPATH**27755 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.27756 **ASYNCHRONOUS EVENTS**

27757 Default.

27758 **STDOUT**

27759 Not used.

**27760 STDERR**

27761           Used only for diagnostic messages.

**27762 OUTPUT FILES**

27763           None.

**27764 EXTENDED DESCRIPTION**

27765           None.

**27766 EXIT STATUS**

27767           The following exit values are returned:

27768           0   Successful completion.

27769           >0  One of the *alias-name* operands specified did not represent a valid alias definition, or an  
27770           error occurred.

**27771 CONSEQUENCES OF ERRORS**

27772           Default.

**27773 APPLICATION USAGE**

27774           Since *unalias* affects the current shell execution environment, it is generally provided as a shell  
27775           regular built-in.

**27776 EXAMPLES**

27777           None.

**27778 FUTURE DIRECTIONS**

27779           None.

**27780 SEE ALSO**

27781           *alias*.

**27782 CHANGE HISTORY**

27783           First released in Issue 4.

27784 **NAME**27785        **uname** — return system name27786 **SYNOPSIS**27787        **uname** [**snrvma**]27788 **DESCRIPTION**

27789        By default, the *uname* utility will write the operating system name to standard output. When  
 27790        options are specified, symbols representing one or more system characteristics will be written to  
 27791        the standard output. The format and contents of the symbols are implementation-dependent.  
 27792        On systems conforming to the **XSH** specification, the symbols written will be those supported  
 27793        by the **XSH** specification *uname()* function.

27794 **OPTIONS**27795        The *uname* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

27796        The following options are supported:

27797        **-a**        Behave as though all of the options **-mnrsv** were specified.

27798        **-m**        Write the name of the hardware type on which the system is running to standard  
 27799        output.

27800        **-n**        Write the name of this node within an implementation-dependent communications  
 27801        network.

27802        **-r**        Write the current release level of the operating system implementation.27803        **-s**        Write the name of the implementation of the operating system.27804        **-v**        Write the current version level of this release of the operating system implementation.

27805        If no options are specified, the *uname* utility will write the operating system name, as if the **-s**  
 27806        option had been specified.

27807 **OPERANDS**

27808        None.

27809 **STDIN**

27810        Not used.

27811 **INPUT FILES**

27812        None.

27813 **ENVIRONMENT VARIABLES**27814        The following environment variables affect the execution of *uname*:

27815        **LANG**     Provide a default value for the internationalisation variables that are unset or null. If  
 27816        **LANG** is unset or null, the corresponding value from the implementation-dependent  
 27817        default locale will be used. If any of the internationalisation variables contains an  
 27818        invalid setting, the utility will behave as if none of the variables had been defined.

27819        **LC\_ALL**

27820        If set to a non-empty string value, override the values of all the other  
 27821        internationalisation variables.

27822        **LC\_CTYPE**

27823        Determine the locale for the interpretation of sequences of bytes of text data as  
 27824        characters (for example, single- as opposed to multi-byte characters in arguments).

27825 **LC\_MESSAGES**  
 27826 Determine the locale that should be used to affect the format and contents of diagnostic  
 27827 messages written to standard error.

27828 EX **NLSPATH**  
 27829 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

27830 **ASYNCHRONOUS EVENTS**  
 27831 Default.

27832 **STDOUT**  
 27833 By default, the output will be a single line of the following form:  
 27834 "%s\n", <sysname>  
 27835 If the **-a** option is specified, the output will be a single line of the following form:  
 27836 "%s %s %s %s %s\n", <sysname>, <nodename>, <release>, <version>,  
 27837 <machine>  
 27838 Additional implementation-dependent symbols may be written; all such symbols will be written  
 27839 at the end of the line of output before the newline character.  
 27840 If options are specified to select different combinations of the symbols, only those symbols will  
 27841 be written, in the order shown above for the **-a** option. If a symbol is not selected for writing, its  
 27842 corresponding trailing blank characters also will not be written.

27843 **STDERR**  
 27844 Used only for diagnostic messages.

27845 **OUTPUT FILES**  
 27846 None.

27847 **EXTENDED DESCRIPTION**  
 27848 None.

27849 **EXIT STATUS**  
 27850 The following exit values are returned:  
 27851 0 The requested information was successfully written.  
 27852 >0 An error occurred.

27853 **CONSEQUENCES OF ERRORS**  
 27854 Default.

27855 **APPLICATION USAGE**  
 27856 Note that any of the symbols could include embedded space characters, which may affect  
 27857 parsing algorithms if multiple options are selected for output.  
 27858 The node name is typically a name that the system uses to identify itself for intersystem  
 27859 communication addressing.

27860 **EXAMPLES**  
 27861 The following command:  
 27862 `uname -sr`  
 27863 writes the operating system name and release level, separated by one or more blank characters.

27864 **FUTURE DIRECTIONS**  
 27865 None.

27866 **SEE ALSO**

27867           The **XSH** specification description of *uname()*.

27868 **CHANGE HISTORY**

27869           First released in Issue 2.

27870 **Issue 4**

27871           Aligned with the ISO/IEC 9945-2: 1993 standard.

27872 **Issue 4, Version 2**

27873           The SYNOPSIS section lists all the valid options.



27874 **NAME**

27875 uncompress — expand compressed data

27876 **SYNOPSIS**27877 EX uncompress [-cfv][*file...*]27878 **DESCRIPTION**

27879 The *uncompress* utility will restore files to their original state after they have been compressed  
 27880 using the *compress* utility. If no files are specified, the standard input will be uncompressed to  
 27881 the standard output. If the invoking process has appropriate privileges, the ownership, modes,  
 27882 access time, and modification time of the original file are preserved.

27883 This utility supports the uncompressing of any files produced by the *compress* utility on the same  
 27884 implementation. For files produced by *compress* on other systems, *uncompress* supports 9- to 14-  
 27885 bit compression (see *compress* -b); it is implementation-dependent whether values of -b greater  
 27886 than 14 are supported.

27887 **OPTIONS**27888 The *uncompress* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**.

27889 The following options are supported:

- 27890 -c Write to standard output; no files will be changed.
- 27891 -f Do not prompt for overwriting files. Except when run in the background, if -f is not  
 27892 given the user will be prompted as to whether an existing file should be overwritten. If  
 27893 the standard input is not a terminal and -f is not given, *uncompress* will write a  
 27894 diagnostic message to standard error and exit with a status greater than zero.
- 27895 -v Write messages to standard error concerning the expansion of each file.

27896 **OPERANDS**

27897 The following operand is supported:

- 27898 *file* A pathname of a file. If *file* already has the .Z suffix specified, it will be used as the  
 27899 input file and the output file will be named *file* with the .Z suffix removed. Otherwise,  
 27900 *file* will be used as the name of the output file and *file* with the .Z suffix appended will  
 27901 be used as the input file.

27902 **STDIN**27903 The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".27904 **INPUT FILES**27905 Input files are in the format produced by the *compress* utility.27906 **ENVIRONMENT VARIABLES**27907 The following environment variables affect the execution of *uncompress*:

- 27908 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 27909 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 27910 default locale will be used. If any of the internationalisation variables contains an  
 27911 invalid setting, the utility will behave as if none of the variables had been defined.

27912 **LC\_ALL**

27913 If set to a non-empty string value, override the values of all the other  
 27914 internationalisation variables.

27915 **LC\_CTYPE**

27916 Determine the locale for the interpretation of sequences of bytes of text data as  
 27917 characters (for example, single- as opposed to multi-byte characters in arguments).

27918 *LC\_MESSAGES*

27919 Determine the locale that should be used to affect the format and contents of diagnostic  
27920 messages written to standard error.

27921 *NLSPATH*

27922 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

27923 **ASYNCHRONOUS EVENTS**

27924 Default.

27925 **STDOUT**

27926 When there are no file operands or the *-c* option is specified, the uncompressed output will be  
27927 written to standard output.

27928 **STDERR**

27929 Prompts will be written to the standard error output under the conditions specified in the  
27930 DESCRIPTION and OPTIONS sections. The prompts will contain the *file* pathname, but their  
27931 format is otherwise unspecified. Otherwise, the standard error output will be used only for  
27932 diagnostic messages.

27933 **OUTPUT FILES**

27934 Output files are the same as the respective input files to *compress*.

27935 **EXTENDED DESCRIPTION**

27936 None.

27937 **EXIT STATUS**

27938 The following exit values are returned:

27939 0 Successful completion.

27940 >0 An error occurred.

27941 **CONSEQUENCES OF ERRORS**

27942 The input file will remain unmodified.

27943 **APPLICATION USAGE**

27944 The limit of 14 on the *compress -b bits* argument is to achieve portability to all systems (within  
27945 the restrictions imposed by the lack of an explicit published file format). Some systems based on  
27946 16-bit architectures cannot support 15- or 16-bit uncompression.

27947 **EXAMPLES**

27948 None.

27949 **FUTURE DIRECTIONS**

27950 None.

27951 **SEE ALSO**

27952 *compress, unpack, zcat*.

27953 **CHANGE HISTORY**

27954 First released in Issue 4.

27955 **Issue 4, Version 2**

27956 The DESCRIPTION section is clarified to state that the ownership, modes, access time, and  
27957 modification time of the original file are preserved if the invoking process has appropriate  
27958 privileges.

27959 **NAME**

27960 unexpand — convert spaces to tabs

27961 **SYNOPSIS**27962 unexpand [ -a | -t *tablist* ][*file...*]27963 **DESCRIPTION**

27964 The *unexpand* utility copies files or standard input to standard output, converting blank  
 27965 characters at the beginning of each line into the maximum number of tab characters followed by  
 27966 the minimum number of space characters needed to fill the same column positions originally  
 27967 filled by the translated blank characters. By default, tabstops are set at every eighth column  
 27968 position. Each backspace character is copied to the output, and causes the column position  
 27969 count for tab calculations to be decremented; the count will never be decremented to a value less  
 27970 than one.

27971 **OPTIONS**27972 The *unexpand* utility supports the **XBD specification, Section 10.2, Utility Syntax Guidelines**.

27973 The following option is supported:

27974 **-a** In addition to translating blank characters at the beginning of each line, translate all  
 27975 sequences of two or more blank characters immediately preceding a tab stop to the  
 27976 maximum number of tab characters followed by the minimum number of space  
 27977 characters needed to fill the same column positions originally filled by the translated  
 27978 blank characters.

27979 **-t *tablist***

27980 Specify the tab stops. The option-argument *tablist* must be a single argument  
 27981 consisting of a single positive decimal integer or multiple positive decimal integers,  
 27982 separated by blank characters or commas, in ascending order. If a single number is  
 27983 given, tabs will be set *tablist* column positions apart instead of the default 8. If multiple  
 27984 numbers are given, the tabs will be set at those specific column positions.

27985 Each tab-stop position *N* must be an integer value greater than zero, and the list must  
 27986 be in strictly ascending order. This is taken to mean that, from the start of a line of  
 27987 output, tabbing to position *N* will cause the next character output to be in the (*N*+1)th  
 27988 column position on that line. When the **-t** option is not specified, the default is the  
 27989 equivalent of specifying **-t 8** (except for the interaction with **-a**, described below).

27990 No space-to-tab character conversions occur for characters at positions beyond the last  
 27991 of those specified in a multiple tab-stop list.

27992 When **-t** is specified, the presence or absence of the **-a** option is ignored; conversion  
 27993 will not be limited to the processing of leading blank characters.

27994 **OPERANDS**

27995 The following operand is supported:

27996 ***file*** A pathname of a text file to be used as input.27997 **STDIN**

27998 See the INPUT FILES section.

27999 **INPUT FILES**

28000 The input files must be text files.

28001 **ENVIRONMENT VARIABLES**

28002 The following environment variables affect the execution of *unexpand*:

28003 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 28004 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 28005 default locale will be used. If any of the internationalisation variables contains an  
 28006 invalid setting, the utility will behave as if none of the variables had been defined.

28007 **LC\_ALL**

28008 If set to a non-empty string value, override the values of all the other  
 28009 internationalisation variables.

28010 **LC\_CTYPE**

28011 Determine the locale for the interpretation of sequences of bytes of text data as  
 28012 characters (for example, single- as opposed to multi-byte characters in arguments and  
 28013 input files), the processing of tab and space characters and for the determination of the  
 28014 width in column positions each character would occupy on a constant-width-font  
 28015 output device.

28016 **LC\_MESSAGES**

28017 Determine the locale that should be used to affect the format and contents of diagnostic  
 28018 messages written to standard error.

28019 EX **NLSPATH**

28020 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28021 **ASYNCHRONOUS EVENTS**

28022 Default.

28023 **STDOUT**

28024 The standard output is equivalent to the input files with the specified space to tab character  
 28025 conversions.

28026 **STDERR**

28027 Used only for diagnostic messages.

28028 **OUTPUT FILES**

28029 None.

28030 **EXTENDED DESCRIPTION**

28031 None.

28032 **EXIT STATUS**

28033 The following exit values are returned:

28034 0 Successful completion.

28035 >0 An error occurred.

28036 **CONSEQUENCES OF ERRORS**

28037 Default.

28038 **APPLICATION USAGE**

28039 One non-intuitive aspect of *unexpand* is its restriction to leading spaces when neither **-a** nor **-t** is  
 28040 specified. Users who desire to always convert all spaces in a file can easily alias *unexpand* to use  
 28041 the **-a** or **-t 8** option.

28042 **EXAMPLES**

28043 None.

28044 **FUTURE DIRECTIONS**

28045           None.

28046 **SEE ALSO**28047           *expand, tabs.*28048 **CHANGE HISTORY**

28049           First released in Issue 4.

28050 **NAME**

28051 unget — undo a previous get of an SCCS file (**DEVELOPMENT**)

28052 **SYNOPSIS**

28053 EX unget [-ns][-r *SID*] *file...*

28054 **DESCRIPTION**

28055 The *unget* utility reverses the effect of a *get* -e done prior to creating the intended new delta.

28056 **OPTIONS**

28057 The *unget* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**. The  
28058 following options are supported:

28059 -r *SID* Uniquely identify which delta is no longer intended. (This would have been specified  
28060 by *get* as the new delta.) The use of this option is necessary only if two or more  
28061 outstanding *get* commands for editing on the same SCCS file were done by the same  
28062 person (login name).

28063 -s Suppress the writing to standard output of the intended delta's SID.

28064 -n Retain the file that was obtained by *get*, which would normally be removed from the  
28065 current directory.

28066 **OPERANDS**

28067 The following operands are supported:

28068 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, *unget* behaves  
28069 as though each file in the directory were specified as a named file, except that non-  
28070 SCCS files (last component of the pathname does not begin with s.) and unreadable  
28071 files are silently ignored.

28072 If a single instance *file* is specified as -, the standard input is read; each line of the  
28073 standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files  
28074 and unreadable files are silently ignored.

28075 **STDIN**

28076 The standard input is a text file used only when the *file* operand is specified as -. Each line of the  
28077 text file is interpreted as an SCCS pathname.

28078 **INPUT FILES**

28079 Any SCCS files processed are files of an unspecified format.

28080 **ENVIRONMENT VARIABLES**

28081 The following environment variables affect the execution of *unget*:

28082 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
28083 *LANG* is unset or null, the corresponding value from the implementation-dependent  
28084 default locale will be used. If any of the internationalisation variables contains an  
28085 invalid setting, the utility will behave as if none of the variables had been defined.

28086 *LC\_ALL*

28087 If set to a non-empty string value, override the values of all the other  
28088 internationalisation variables.

28089 *LC\_CTYPE*

28090 Determine the locale for the interpretation of sequences of bytes of text data as  
28091 characters (for example, single- as opposed to multi-byte characters in arguments and  
28092 input files).

28093 **LC\_MESSAGES**  
 28094 Determine the locale that should be used to affect the format and contents of diagnostic  
 28095 messages written to standard error.

28096 **NLSPATH**  
 28097 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28098 **ASYNCHRONOUS EVENTS**  
 28099 Default.

28100 **STDOUT**  
 28101 The standard output consists of a line for each file, in the following format:  
 28102 "%s\n", <SID removed from file>  
 28103 If there is more than one named file or if a directory or standard input is named, each pathname  
 28104 is written before each of the preceding lines:  
 28105 "\n%s:\n", <pathname>

28106 **STDERR**  
 28107 Used only for diagnostic messages.

28108 **OUTPUT FILES**  
 28109 Any SCCS files updated are files of an unspecified format. During processing of a *file*, a locking  
 28110 *z-file*, as described in *get*, and a *q-file* (a working copy of the *p-file*), may be created and deleted.  
 28111 The *p-file* and *g-file*, as described in *get*, are deleted.

28112 **EXTENDED DESCRIPTION**  
 28113 None.

28114 **EXIT STATUS**  
 28115 The following exit values are returned:  
 28116 0 Successful completion.  
 28117 >0 An error occurred.

28118 **CONSEQUENCES OF ERRORS**  
 28119 Default.

28120 **APPLICATION USAGE**  
 28121 None.

28122 **EXAMPLES**  
 28123 None.

28124 **FUTURE DIRECTIONS**  
 28125 None.

28126 **SEE ALSO**  
 28127 *delta*, *get*, *sact*.

28128 **CHANGE HISTORY**

28129 First released in Issue 2.

28130 **Issue 4**

28131 Format reorganised.

28132 Utility Syntax Guidelines support mandated.

28133 Internationalised environment variable support mandated.



28134 **NAME**28135           **uniq** — report or filter out repeated lines in a file28136 **SYNOPSIS**28137           **uniq** [-c|-d|-u][-f *fields*][-s *char*][*input\_file* [*output\_file*]]28138 OB       **uniq** [-c|-d|-u][-n][+m][*input\_file* [*output\_file*]]28139 **DESCRIPTION**

28140       The *uniq* utility will read an input file comparing adjacent lines, and write one copy of each input  
 28141       line on the output. The second and succeeding copies of repeated adjacent input lines will not  
 28142       be written.

28143       Repeated lines in the input will not be detected if they are not adjacent.

28144 **OPTIONS**

28145 OB       The *uniq* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines**; the  
 28146       obsolescent version does not, as one of the options begins with "+" and the *-m* and *+n* options do  
 28147       not have option letters.

28148       The following options are supported:

28149       **-c**       Precede each output line with a count of the number of times the line occurred in the  
 28150       input.

28151       **-d**       Suppress the writing of lines that are not repeated in the input.28152       **-f *fields***

28153       Ignore the first *fields* fields on each input line when doing comparisons, where *fields* is a  
 28154       positive decimal integer. A field is the maximal string matched by the basic regular  
 28155       expression:

28156           [[ :blank: ]]\*[ ^[ :blank: ] ]\*

28157       If the *fields* option-argument specifies more fields than appear on an input line, a null  
 28158       string will be used for comparison.

28159       **-s *chars***

28160       Ignore the first *chars* characters when doing comparisons, where *chars* is a positive  
 28161       decimal integer. If specified in conjunction with the *-f* option, the first *chars* characters  
 28162       after the first *fields* fields will be ignored. If the *chars* option-argument specifies more  
 28163       characters than remain on an input line, a null string will be used for comparison.

28164       **-u**       Suppress the writing of lines that are repeated in the input.28165 OB       **-n**       Equivalent to *-f fields* with *fields* set to *n*.28166 OB       **+m**       Equivalent to *-s chars* with *chars* set to *m*.28167 **OPERANDS**

28168       The following operands are supported:

28169       *input\_file*

28170       A pathname of the input file. If the *input\_file* operand is not specified, or if the *input\_file*  
 28171       is "-", the standard input will be used.

28172       *output\_file*

28173       A pathname of the output file. If the *output\_file* operand is not specified, the standard  
 28174       output will be used. The results are unspecified if the file named by *output\_file* is the  
 28175       file named by *input\_file*.

28176 **STDIN**

28177 The standard input will be used only if no *input\_file* operand is specified or if *input\_file* is "-". See  
28178 the INPUT FILES section.

28179 **INPUT FILES**

28180 The input file must be a text file.

28181 **ENVIRONMENT VARIABLES**

28182 The following environment variables affect the execution of *uniq*:

28183 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
28184 *LANG* is unset or null, the corresponding value from the implementation-dependent  
28185 default locale will be used. If any of the internationalisation variables contains an  
28186 invalid setting, the utility will behave as if none of the variables had been defined.

28187 *LC\_ALL*

28188 If set to a non-empty string value, override the values of all the other  
28189 internationalisation variables.

28190 *LC\_CTYPE*

28191 Determine the locale for the interpretation of sequences of bytes of text data as  
28192 characters (for example, single- as opposed to multi-byte characters in arguments and  
28193 input files) which characters constitute a blank character in the current locale.

28194 *LC\_MESSAGES*

28195 Determine the locale that should be used to affect the format and contents of diagnostic  
28196 messages written to standard error.

28197 EX *NLSPATH*

28198 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28199 **ASYNCHRONOUS EVENTS**

28200 Default.

28201 **STDOUT**

28202 The standard output will be used only if no *output\_file* operand is specified. See the OUTPUT  
28203 FILES section.

28204 **STDERR**

28205 Used only for diagnostic messages.

28206 **OUTPUT FILES**

28207 If the *-c* option is specified, the output file must be empty or each line must be of the form:

28208 "%d %s", *<number of duplicates>*, *<line>*

28209 otherwise, the output file must be empty or each line must be of the form:

28210 "%s", *<line>*

28211 **EXTENDED DESCRIPTION**

28212 None.

28213 **EXIT STATUS**

28214 The following exit values are returned:

28215 0 The utility executed successfully.

28216 >0 An error occurred.

28217 **CONSEQUENCES OF ERRORS**

28218 Default.

28219 **APPLICATION USAGE**

28220 The *sort* utility can be used to cause repeated lines to be adjacent in the input file.

28221 **EXAMPLES**

28222 The following input file data (but flushed left) was used for a test series on *uniq*:

```
28223 #01 foo0 bar0 fool bar1
28224 #02 bar0 fool bar1 fool
28225 #03 foo0 bar0 fool bar1
28226 #04
28227 #05 foo0 bar0 fool bar1
28228 #06 foo0 bar0 fool bar1
28229 #07 bar0 fool bar1 foo0
```

28230 What follows is a series of test invocations of the *uniq* utility that use a mixture of *uniq* options  
 28231 against the input file data. These tests verify the meaning of *adjacent*. The *uniq* utility views the  
 28232 input data as a sequence of strings delimited by \n. Accordingly, for the *fields* member of the  
 28233 sequence, *uniq* interprets unique or repeated adjacent lines strictly relative to the *fields*+1th  
 28234 member.

- 28235 1. This first example tests the line counting option, comparing each line of the input file data  
 28236 starting from the second field:

```
28237 uniq -c -f 1 uniq_0I.t
28238 1 #01 foo0 bar0 fool bar1
28239 1 #02 bar0 fool bar1 foo0
28240 1 #03 foo0 bar0 fool bar1
28241 1 #04
28242 2 #05 foo0 bar0 fool bar1
28243 1 #07 bar0 fool bar1 foo0
```

28244 The number 2, prefixing the fifth line of output, signifies that the *uniq* utility detected a  
 28245 pair of repeated lines. Given the input data, this can only be true when *uniq* is run using  
 28246 the *-f 1* option (which causes *uniq* to ignore the first field on each input line).

- 28247 2. The second example tests the option to suppress unique lines, comparing each line of the  
 28248 input file data starting from the second field:

```
28249 uniq -d -f 1 uniq_0I.t
28250 #05 foo0 bar0 fool bar1
```

- 28251 3. This test suppresses repeated lines, comparing each line of the input file data starting from  
 28252 the second field:

```
28253 uniq -u -f 1 uniq_0I.t
28254 #01 foo0 bar0 fool bar1
28255 #02 bar0 fool bar1 fool
28256 #03 foo0 bar0 fool bar1
28257 #04
28258 #07 bar0 fool bar1 foo0
```

- 28259 4. This suppresses unique lines, comparing each line of the input file data starting from the  
 28260 third character:

```
28261 uniq -d -s 2 uniq_0I.t
```

28262 In the last example, the *uniq* utility found no input matching the above criteria.

28263 **FUTURE DIRECTIONS**

28264           None.

28265 **SEE ALSO**

28266           *comm, sort.*

28267 **CHANGE HISTORY**

28268           First released in Issue 2.

28269 **Issue 4**

28270           Aligned with the ISO/IEC 9945-2: 1993 standard.

|  
|

28271 **NAME**28272           unlink — call the *unlink()* function28273 **SYNOPSIS**28274 EX       unlink *file*

28275

28276 **DESCRIPTION**28277           The *unlink* utility performs the function call:28278                 unlink(*file*);28279           A user may need appropriate privilege to invoke the *unlink* utility.28280 **OPTIONS**

28281           None.

28282 **OPERANDS**

28283           The following operands are supported:

28284         *file*       The pathname of an existing file.28285 **STDIN**

28286           Not used.

28287 **INPUT FILES**

28288           Not used.

28289 **ENVIRONMENT VARIABLES**28290           The following environment variables affect the execution of *unlink*:

28291         *LANG*   Provide a default value for the internationalization variables that are unset or null. If  
 28292           *LANG* is unset or null, the corresponding value from the implementation-dependent  
 28293           default locale will be used. If any of the internationalization variables contain an  
 28294           invalid setting, the utility will behave as if none of the variables had been set.

28295         *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 28296           internationalization variables.

28297         *LC\_CTYPE*

28298           Determine the locale for the interpretation of sequences of bytes of text data as  
 28299           characters (for example, single- as opposed to multi-byte characters in arguments).

28300         *LC\_MESSAGES*

28301           Determine the locale that should be used to affect the format and contents of diagnostic  
 28302           messages written to standard error.

28303         *NLSPATH*28304           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.28305 **ASYNCHRONOUS EVENTS**

28306           Default.

28307 **STDOUT**

28308           None.

28309 **STDERR**

28310           Used only for diagnostic messages.

28311 **OUTPUT FILES**

28312           None.

28313 **EXTENDED DESCRIPTION**

28314           None.

28315 **EXIT STATUS**

28316           The following exit values are returned:

28317           0   Successful completion.

28318           >0  An error occurred.

28319 **CONSEQUENCES OF ERRORS**

28320           Default.

28321 **APPLICATION USAGE**

28322           None.

28323 **EXAMPLES**

28324           None.

28325 **FUTURE DIRECTIONS**

28326           None.

28327 **SEE ALSO**

28328           *link*, *rm*, the **XSH** specification description of *unlink()*.

28329 **CHANGE HISTORY**

28330           First released in Issue 5.

28331 **NAME**28332           unpack — expand files (**LEGACY**)28333 **SYNOPSIS**28334 EX       unpack *file...*28335 **DESCRIPTION**

28336       The *unpack* utility replaces files in the format used by *pack* with their unpacked form. For each  
 28337       file *file* operand, a search is made for a file called *file.z* (or just *file*, if *file* ends in *.z*). If this file  
 28338       appears to be a packed file, it is replaced by its expanded version. The new file has the *.z*  
 28339       stripped from its name. If the invoking process has appropriate privileges, the ownership,  
 28340       modes, access time, and modification time of the original file are preserved.

28341       A file is not unpacked if one of the following is true:

- 28342           • The filename (exclusive of the *.z*) has more than {NAME\_MAX} bytes.
- 28343           • The file cannot be opened.
- 28344           • The file does not appear to be the output of *pack*.
- 28345           • A file with the unpacked name already exists.
- 28346           • The unpacked file cannot be created.

28347 **OPTIONS**

28348       None.

28349 **OPERANDS**

28350       The following operand is supported:

28351       *file*       A pathname of a file to be unpacked; *file* can include or omit the *.z* suffix.

28352 **STDIN**

28353       Not used.

28354 **INPUT FILES**28355       The input files are regular files in the format created by *pack*.28356 **ENVIRONMENT VARIABLES**28357       The following environment variables may affect the execution of *pack*:

28358       *LANG*       Provide a default value for the internationalisation variables that are unset or null. If  
 28359       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 28360       default locale will be used. If any of the internationalisation variables contains an  
 28361       invalid setting, the utility will behave as if none of the variables had been defined.

28362       *LC\_ALL*

28363           If set to a non-empty string value, override the values of all the other  
 28364       internationalisation variables.

28365       *LC\_CTYPE*

28366           Determine the locale for the interpretation of sequences of bytes of text data as  
 28367       characters (for example, single- as opposed to multi-byte characters in arguments).

28368       *LC\_MESSAGES*

28369           Determine the locale that should be used to affect the format and contents of diagnostic  
 28370       messages written to standard error, and informative messages written to standard  
 28371       output.

28372       *NLSPATH*

28373           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28374 **ASYNCHRONOUS EVENTS**

28375 If an error occurs, the \$(basename file .z) file is not created and the original file is unchanged.

28376 **STDOUT**

28377 The standard output is a text file containing one line for each file unpacked, with the following  
28378 format in the POSIX locale:

28379 "unpack: %s: unpacked\n", file

28380 **STDERR**

28381 Used only for diagnostic messages.

28382 **OUTPUT FILES**

28383 Files equivalent to the original unpacked file are created with the names as though  
28384 \$(basename file .z). were invoked corresponding to each file operand.

28385 **EXTENDED DESCRIPTION**

28386 None.

28387 **EXIT STATUS**

28388 The following exit values are returned:

28389 0 Successful completion.

28390 >0 An error occurred.

28391 **CONSEQUENCES OF ERRORS**

28392 Default.

28393 **APPLICATION USAGE**

28394 Applications should migrate to the *uncompress* utility.

28395 **EXAMPLES**

28396 None.

28397 **FUTURE DIRECTIONS**

28398 None.

28399 **SEE ALSO**

28400 *pack*, *pcat*, *uncompress*.

28401 **CHANGE HISTORY**

28402 First released in Issue 2.

28403 **Issue 4**

28404 Format reorganised.

28405 Split into a separate description.

28406 Marked TO BE WITHDRAWN.

28407 Internationalised environment variable support made optional.

28408 **Issue 4, Version 2**

28409 The DESCRIPTION section is revised as follows:

28410 • It states that the ownership, modes, access time and modification time of the original file are  
28411 preserved if the invoking process has appropriate privileges.

28412 • The assertion that a file is not unpacked if the filename has more than {NAME\_MAX}-2  
28413 bytes now specifies {NAME\_MAX} bytes.



28414 **Issue 5**

28415 Marked LEGACY.

|

28416 **NAME**

28417           uucp — system-to-system copy

28418 **SYNOPSIS**28419 UN EX   uucp [-cCdfjmr][*-n user*] *source-file... destination-file*28420 **DESCRIPTION**28421           The *uucp* utility copies files named by the *source-file* arguments to the *destination-file* argument.  
28422           The files named can be on local or remote systems.28423           The *uucp* utility cannot guarantee support for all character encodings in all circumstances. For  
28424           example, transmission data may be restricted to 7-bits by the underlying network, 8-bit data and  
28425           filenames need not be portable to non-internationalised systems, and so on. Under these  
28426           circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
28427           standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used,  
28428           and that only characters defined in the Portable Filename Character Set be used for naming files.28429 **OPTIONS**28430           The *uucp* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
28431           following options are supported:28432           **-c**       Do not copy local file to the spool directory for transfer to the remote machine (default).28433 UN       **-C**       Force the copy of local files to the spool directory for transfer.28434           **-d**       Make all necessary directories for the file copy (default).28435 UN       **-f**       Do not make intermediate directories for the file copy.28436 UN       **-j**       Write the job identification string to standard output. This job identification can be  
28437           used by *uustat* to obtain the status or terminate a job.28438           **-m**       Send mail to the requester when the copy is completed.28439 UN       **-n user**  
28440           Notify *user* on the remote system that a file was sent.28441 UN       **-r**       Do not start the file transfer; just queue the job.28442 **OPERANDS**

28443           The following operands are supported:

28444           *destination-file*28445           *source-file*28446           A pathname of a file to be copied to, or from, respectively. Either name can be a  
28447           pathname on the local machine, or can have the form:28448                 *system-name!pathname*28449           where *system-name* is taken from a list of system names that *uucp* knows about; see  
28450           *uname*. The destination *system-name* can also be a list of names such as:28451                 *system-name!system-name!...!system-name!pathname*28452           in which case, an attempt is made to send the file via the specified route to the  
28453           destination. Care should be taken to ensure that intermediate nodes in the route are  
28454           willing to forward information.28455           The shell pattern matching notation characters "?", "\*" and [...] appearing in *pathname*  
28456           will be expanded on the appropriate system.

28457 Pathnames can be one of:

- 28458 1. An absolute pathname.
- 28459 2. A pathname preceded by `~user` where *user* is a login name on the specified system  
 28460 and is replaced by that user's login directory. Note that if an invalid login is  
 28461 specified, the default is to the public directory (called "*PUBDIR*"; the actual  
 28462 location of *PUBDIR* is implementation-dependent).
- 28463 3. A pathname preceded by `~/destination` where *destination* is appended to *PUBDIR*.
- 28464 **Note:** This destination will be treated as a filename unless more than one file is  
 28465 being transferred by this request or the destination is already a directory.  
 28466 To ensure that it is a directory, follow the destination with a `/`. For  
 28467 example, `~/dan/` as the destination will make the directory *PUBDIR/dan*  
 28468 if it does not exist and put the requested files in that directory.
- 28469 4. Anything else is prefixed by the current directory.

28470 If the result is an erroneous pathname for the remote system, the copy will fail. If the  
 28471 *destination-file* is a directory, the last part of the *source-file* name is used.

28472 The read, write and execute permissions given by *uucp* are implementation-dependent.

#### 28473 **STDIN**

28474 Not used.

#### 28475 **INPUT FILES**

28476 The files to be copied are regular files.

#### 28477 **ENVIRONMENT VARIABLES**

28478 The following environment variables affect the execution of *uucp*:

28479 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 28480 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 28481 default locale will be used. If any of the internationalisation variables contains an  
 28482 invalid setting, the utility will behave as if none of the variables had been defined.

#### 28483 **LC\_ALL**

28484 If set to a non-empty string value, override the values of all the other  
 28485 internationalisation variables.

#### 28486 **LC\_COLLATE**

28487 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 28488 character collating elements within bracketed filename patterns.

#### 28489 **LC\_CTYPE**

28490 Determine the locale for the interpretation of sequences of bytes of text data as  
 28491 characters (for example, single- as opposed to multi-byte characters in arguments and  
 28492 input files) and the behaviour of character classes within bracketed filename patterns  
 28493 (for example, `'[[:lower:]]*'`).

#### 28494 **LC\_MESSAGES**

28495 Determine the locale that should be used to affect the format and contents of diagnostic  
 28496 messages written to standard error, and informative messages written to standard  
 28497 output.

#### 28498 **LC\_TIME**

28499 Determine the format of date and time strings output by *uucp*.

28500 *NLSPATH*  
 28501 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28502 *TZ* Determine the timezone used with date and time strings.

28503 **ASYNCHRONOUS EVENTS**  
 28504 Default.

28505 **STDOUT**  
 28506 Not used.

28507 **STDERR**  
 28508 Used only for diagnostic messages.

28509 **OUTPUT FILES**  
 28510 The output files (which may be on other systems) are copies of the input files.  
 28511 If the *-m* is used, mail files will be modified.

28512 **EXTENDED DESCRIPTION**  
 28513 None.

28514 **EXIT STATUS**  
 28515 The following exit values are returned:  
 28516 0 Successful completion.  
 28517 >0 An error occurred.

28518 **CONSEQUENCES OF ERRORS**  
 28519 Default.

28520 **APPLICATION USAGE**  
 28521 The domain of remotely accessible files can (and for obvious security reasons usually should) be  
 28522 severely restricted.

28523 Note that the *!"* character in addresses has to be escaped when using *cs**h* as a command  
 28524 interpreter because of its history substitution syntax. For *ksh* and *sh* the escape is not necessary,  
 28525 but may be used.

28526 Typical implementations of this utility require a communications line configured to use the **XBD**  
 28527 specification, **Chapter 9, General Terminal Interface**, but other communications means may be  
 28528 used. On systems where there are no available communications means (either temporarily or  
 28529 permanently), this utility will write an error message describing the problem and exit with a  
 28530 non-zero exit status.

28531 As noted above, shell metacharacters appearing in pathnames are expanded on the appropriate  
 28532 system. On an internationalised system, this is done under the control of local settings of  
 28533 *LC\_COLLATE* and *LC\_CTYPE*. Thus, care should be taken when using bracketed filename  
 28534 patterns, as collation and typing rules may vary from one system to another. Also be aware that  
 28535 certain types of expression (that is, equivalence classes, character classes and collating symbols)  
 28536 need not be supported on non-internationalised systems.

28537 **EXAMPLES**  
 28538 None.

28539 **FUTURE DIRECTIONS**  
 28540 None.

28541 **SEE ALSO**  
 28542 *mailx*, *uuencode*, *uulog*, *uuname*, *uustat*, *uux*.

28543 **CHANGE HISTORY**

28544 First released in Issue 2.

28545 **Issue 4**

28546 Format reorganised.

28547 Split into a separate description.

28548 Utility Syntax Guidelines support mandated.

28549 Internationalised environment variable support mandated.

28550 Presence of the utility mandated, even on systems where no communications are available.

28551 **NAME**

28552           uudecode — decode a binary file

28553 **SYNOPSIS**28554           uudecode [*file*]28555 **DESCRIPTION**

28556           The *uudecode* utility reads a file or standard input if no file is specified, that includes data created  
 28557           by the *uuencode* utility. The *uudecode* utility scans the input file, searching for data compatible  
 28558           with the format specified in *uuencode* and attempts to create or overwrite the file described by  
 28559           the data. The pathname, file access permission bits and contents for the file to be produced are  
 28560           all contained in that data. The mode bits of the created file will be set from the file access  
 28561           permission bits contained in the data; that is, other attributes of the mode, including the file  
 28562           mode creation mask (see *umask*), will not affect the file being produced.

28563           If the pathname of the file to be produced exists, and the user does not have write permission on  
 28564           that file, *uudecode* will terminate with an error. If the pathname of the file to be produced exists,  
 28565           and the user has write permission on that file, the existing file will be overwritten.

28566           If the input data was produced by *uuencode* on a system with a different number of bits per byte  
 28567           than on the target system, the results of *uudecode* are unspecified.

28568 **OPTIONS**

28569           None.

28570 **OPERANDS**

28571           The following operand is supported:

28572           *file*       The pathname of a file containing the output of *uuencode*.28573 **STDIN**

28574           See the INPUT FILES section.

28575 **INPUT FILES**28576           The input files must be files containing the output of *uuencode*.28577 **ENVIRONMENT VARIABLES**28578           The following environment variables affect the execution of *uudecode*:

28579           *LANG*   Provide a default value for the internationalisation variables that are unset or null. If  
 28580           *LANG* is unset or null, the corresponding value from the implementation-dependent  
 28581           default locale will be used. If any of the internationalisation variables contains an  
 28582           invalid setting, the utility will behave as if none of the variables had been defined.

28583           *LC\_ALL*

28584           If set to a non-empty string value, override the values of all the other  
 28585           internationalisation variables.

28586           *LC\_CTYPE*

28587           Determine the locale for the interpretation of sequences of bytes of text data as  
 28588           characters (for example, single- as opposed to multi-byte characters in arguments and  
 28589           input files).

28590           *LC\_MESSAGES*

28591           Determine the locale that should be used to affect the format and contents of diagnostic  
 28592           messages written to standard error.

28593 EX       *NLSPATH*28594           Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

**28595 ASYNCHRONOUS EVENTS**

28596 Default.

**28597 STDOUT**

28598 Not used.

**28599 STDERR**

28600 Used only for diagnostic messages.

**28601 OUTPUT FILES**

28602 The output file will be in the same format as the file originally encoded by *uuencode*.

**28603 EXTENDED DESCRIPTION**

28604 None.

**28605 EXIT STATUS**

28606 The following exit values are returned:

28607 0 Successful completion.

28608 >0 An error occurred.

**28609 CONSEQUENCES OF ERRORS**

28610 Default.

**28611 APPLICATION USAGE**

28612 The user who is invoking *uudecode* must have write permission on any file being created.

28613 The output of *uuencode* is essentially an encoded bit stream that is not cognizant of byte  
28614 boundaries. It is possible that a 9-bit byte target machine can process input from an 8-bit source,  
28615 if it is aware of the requirement, but the reverse is unlikely to be satisfying. Of course, the only  
28616 data that is meaningful for such a transfer between architectures is generally character data.

**28617 EXAMPLES**

28618 None.

**28619 FUTURE DIRECTIONS**

28620 None.

**28621 SEE ALSO**

28622 *uuencode*.

**28623 CHANGE HISTORY**

28624 First released in Issue 4.

28625 **NAME**

28626 uuencode — encode a binary file

28627 **SYNOPSIS**

28628 uuencode [*file*] *decode\_pathname*

28629 **DESCRIPTION**

28630 The *uuencode* utility writes an encoded version of the named input file, or standard input if no  
 28631 *file* is specified, to standard output. The output is encoded using the algorithm described in the  
 28632 STDOUT section and includes the file access permission bits (in *chmod* octal or symbolic  
 28633 notation) of the input file and the *decode\_pathname*, for re-creation of the file on another system  
 28634 that conforms to this specification.

28635 **OPTIONS**

28636 None.

28637 **OPERANDS**

28638 The following operands are supported:

28639 *decode\_pathname*

28640 The pathname of the file into which the *uudecode* utility will place the decoded file. If  
 28641 there are characters in *decode\_pathname* that are not in the portable filename character  
 28642 set the results are unspecified.

28643 *file* A pathname of the file to be encoded.

28644 **STDIN**

28645 See the INPUT FILES section.

28646 **INPUT FILES**

28647 Input files can be files of any type.

28648 **ENVIRONMENT VARIABLES**

28649 The following environment variables affect the execution of *uuencode*:

28650 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 28651 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 28652 default locale will be used. If any of the internationalisation variables contains an  
 28653 invalid setting, the utility will behave as if none of the variables had been defined.

28654 *LC\_ALL*

28655 If set to a non-empty string value, override the values of all the other  
 28656 internationalisation variables.

28657 *LC\_CTYPE*

28658 Determine the locale for the interpretation of sequences of bytes of text data as  
 28659 characters (for example, single- as opposed to multi-byte characters in arguments and  
 28660 input files).

28661 *LC\_MESSAGES*

28662 Determine the locale that should be used to affect the format and contents of diagnostic  
 28663 messages written to standard error.

28664 EX *NLSPATH*

28665 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28666 **ASYNCHRONOUS EVENTS**

28667 Default.



**28668 STDOUT**

28669 The standard output is a text file (encoded in the character set of the current locale) that begins  
 28670 with the line:

28671 `"beginΔ%sΔ%s\n", <mode>, decode_pathname`

28672 and ends with the line:

28673 `end\n`

28674 In both cases, the lines have no preceding or trailing blank characters.

28675 The algorithm that is used for lines in between **begin** and **end** takes three octets as input and  
 28676 writes four characters of output by splitting the input at six-bit intervals into four octets,  
 28677 containing data in the lower six bits only. These octets are converted to characters by adding a  
 28678 value of 0x20 to each octet, so that each octet is in the range 0x20–0x5f, and then it is assumed to  
 28679 represent a printable character in the ISO/IEC 646:1991 standard encoded character set. It then  
 28680 will be translated into the corresponding character codes for the codeset in use in the current  
 28681 locale. (For example, the octet 0x41, representing A, would be translated to A in the current  
 28682 codeset, such as 0xc1 if it were EBCDIC.)

28683 Where the bits of two octets are combined, the least significant bits of the first octet are shifted  
 28684 left and combined with the most significant bits of the second octet shifted right. Thus the three  
 28685 octets A, B, C are converted into the four octets:

```
28686 0x20 + ((A >> 2) & 0x3F)
28687 0x20 + (((A << 4) | ((B >> 4) & 0xF)) & 0x3F)
28688 0x20 + (((B << 2) | ((C >> 6) & 0x3)) & 0x3F)
28689 0x20 + ((C) & 0x3F)
```

28690 These octets are then translated into the local character set.

28691 Each encoded line contains a length character, equal to the number of characters to be decoded  
 28692 plus 0x20 translated to the local character set as described above, followed by the encoded  
 28693 characters. The maximum number of octets to be encoded on each line is 45.

**28694 STDERR**

28695 Used only for diagnostic messages.

**28696 OUTPUT FILES**

28697 None.

**28698 EXTENDED DESCRIPTION**

28699 None.

**28700 EXIT STATUS**

28701 The following exit values are returned:

28702 `0` Successful completion.

28703 `>0` An error occurred.

**28704 CONSEQUENCES OF ERRORS**

28705 Default.

**28706 APPLICATION USAGE**

28707 The file is expanded by 35 percent (each three octets become four, plus control information)  
 28708 causing it to take longer to transmit.

28709 Since this utility is intended to create files to be used for data interchange between systems with  
 28710 possibly different codesets, and to represent binary data as a text file, the ISO/IEC 646:1991  
 28711 standard was chosen for a midpoint in the algorithm as a known reference point. The output

28712 from *uuencode* is a text file on the local system. If the output were in the ISO/IEC 646:1991  
28713 standard codeset, it might not be a text file (at least because the newline characters might not  
28714 match), and the goal of creating a text file would be defeated. If this text file was then carried to  
28715 another machine with the same codeset, it would be perfectly compatible with that system's  
28716 *uudecode*. If it was transmitted over a mail system or sent to a machine with a different codeset,  
28717 it is assumed that, as for every other text file, some translation mechanism would convert it (by  
28718 the time it reached a user on the other system) into an appropriate codeset. This translation only  
28719 makes sense from the local codeset, not if the file has been put into a ISO/IEC 646:1991 standard  
28720 representation first. Similarly, files processed by *uuencode* can be placed in *pax* archives,  
28721 intermixed with other text files in the same codeset.

28722 The algorithm is described in terms of 8-bit quantities, or octets. Since no byte alignment is  
28723 implied, it will encode data from machines with any number of bits per byte. However, unless  
28724 that encoded data is then decoded on a machine with the same number of bits per byte, the  
28725 output might not be useful.

28726 **EXAMPLES**

28727 None.

28728 **FUTURE DIRECTIONS**

28729 None.

28730 **SEE ALSO**

28731 *mailx*, *uudecode*.

28732 **CHANGE HISTORY**

28733 First released in Issue 4.

28734 **NAME**28735 uulog — query system-to-system transaction log (**LEGACY**)28736 **SYNOPSIS**28737 UN EX uulog [-s *system*]28738 **DESCRIPTION**28739 The *uulog* utility writes the status of *uucp* or *uux* transactions to standard output.28740 **OPTIONS**28741 The *uulog* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
28742 following option is supported:28743 -s *system*28744 Write information about file transfer work involving system *system*. By default,  
28745 information is written about all systems.28746 **OPERANDS**

28747 None.

28748 **STDIN**

28749 Not used.

28750 **INPUT FILES**

28751 None.

28752 **ENVIRONMENT VARIABLES**28753 The following environment variables affect the execution of *uulog*:28754 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
28755 *LANG* is unset or null, the corresponding value from the implementation-dependent  
28756 default locale will be used. If any of the internationalisation variables contains an  
28757 invalid setting, the utility will behave as if none of the variables had been defined.28758 **LC\_ALL**28759 If set to a non-empty string value, override the values of all the other  
28760 internationalisation variables.28761 **LC\_CTYPE**28762 Determine the locale for the interpretation of sequences of bytes of text data as  
28763 characters (for example, single- as opposed to multi-byte characters in arguments and  
28764 input files).28765 **LC\_MESSAGES**28766 Determine the locale that should be used to affect the format and contents of diagnostic  
28767 messages written to standard error, and informative messages written to standard  
28768 output.28769 **LC\_TIME**28770 Determine the format of date and time strings output by *uulog*.28771 **NLSPATH**28772 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.28773 **TZ** Determine the timezone used with date and time strings.28774 **ASYNCHRONOUS EVENTS**

28775 Default.

**28776 STDOUT**

28777 The standard output is a text file indicating the status of file transfer work. The format is  
28778 unspecified.

**28779 STDERR**

28780 Used only for diagnostic messages.

**28781 OUTPUT FILES**

28782 None.

**28783 EXTENDED DESCRIPTION**

28784 None.

**28785 EXIT STATUS**

28786 The following exit values are returned:

28787 0 Successful completion.

28788 >0 An error occurred.

**28789 CONSEQUENCES OF ERRORS**

28790 Default.

**28791 APPLICATION USAGE**

28792 None.

**28793 EXAMPLES**

28794 None.

**28795 FUTURE DIRECTIONS**

28796 None.

**28797 SEE ALSO**

28798 *uucp, uuname, uustat, uux.*

**28799 CHANGE HISTORY**

28800 First released in Issue 2. |

**28801 Issue 4**

28802 Format reorganised.

28803 Split into a separate description.

28804 Utility Syntax Guidelines support mandated.

28805 Internationalised environment variable support mandated. |

**28806 Issue 5**

28807 Marked LEGACY. |

28808 **NAME**28809 uuname — list names of other known uucp systems (**LEGACY**)28810 **SYNOPSIS**

28811 UN EX uuname [-l]

28812 **DESCRIPTION**28813 The *uuname* utility writes the *uucp* names of known systems, one per line.28814 **OPTIONS**28815 The *uuname* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
28816 following option is supported:

28817 -l Write the local system name to standard output.

28818 **OPERANDS**

28819 None.

28820 **STDIN**

28821 Not used.

28822 **INPUT FILES**

28823 None.

28824 **ENVIRONMENT VARIABLES**28825 The following environment variables affect the execution of *uuname*:28826 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
28827 *LANG* is unset or null, the corresponding value from the implementation-dependent  
28828 default locale will be used. If any of the internationalisation variables contains an  
28829 invalid setting, the utility will behave as if none of the variables had been defined.28830 *LC\_ALL*28831 If set to a non-empty string value, override the values of all the other  
28832 internationalisation variables.28833 *LC\_CTYPE*28834 Determine the locale for the interpretation of sequences of bytes of text data as  
28835 characters (for example, single- as opposed to multi-byte characters in arguments).28836 *LC\_MESSAGES*28837 Determine the locale that should be used to affect the format and contents of diagnostic  
28838 messages written to standard error, and informative messages written to standard  
28839 output.28840 *NLSPATH*28841 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.28842 **ASYNCHRONOUS EVENTS**

28843 Default.

28844 **STDOUT**

28845 The standard output is a text file listing names of systems, in the following format:

28846 "%s\n", &lt;system name&gt;

28847 **STDERR**

28848 Used only for diagnostic messages.

28849 **OUTPUT FILES**

28850 None.

28851 **EXTENDED DESCRIPTION**

28852       None.

28853 **EXIT STATUS**

28854       The following exit values are returned:

28855       0   Successful completion.

28856       &gt;0   An error occurred.

28857 **CONSEQUENCES OF ERRORS**

28858       Default.

28859 **APPLICATION USAGE**

28860       None.

28861 **EXAMPLES**

28862       None.

28863 **FUTURE DIRECTIONS**

28864       None.

28865 **SEE ALSO**28866       *uucp, uulog, uustat, uux.*28867 **CHANGE HISTORY**

28868       First released in Issue 2. |

28869 **Issue 4**

28870       Format reorganised.

28871       Split into a separate description.

28872       Utility Syntax Guidelines support mandated.

28873       Internationalised environment variable support mandated. |

28874 **Issue 5** |

28875       Marked LEGACY.

28876 **NAME**28877 uupick — receive public system-to-system file copies (**LEGACY**)28878 **SYNOPSIS**28879 UN EX uupick [-s *system*]28880 **DESCRIPTION**

28881 The *uupick* utility can be used by a user to accept or reject the files transmitted to the user.  
 28882 Specifically, *uupick* searches the public directory (called “*PUBDIR*”; the actual location of  
 28883 *PUBDIR* is implementation-dependent) on the user’s system for files sent to the user. For each  
 28884 entry (file or directory) found, the user is prompted for each file or directory. The *uupick* utility  
 28885 then reads a line from the standard input to determine the disposition of the file. The user’s  
 28886 possible responses are:

28887 newline Go on to next entry.

28888 **d** Delete the entry.

28889 **m**[*dir*] Move the entry to named directory *dir*. If *dir* is not specified as an absolute pathname a  
 28890 destination relative to the current directory is assumed. If no destination is given, the  
 28891 default is the current directory.

28892 **a**[*dir*] Same as m except moving all the files sent from *system*.28893 **p** Write the content of the file to standard output.28894 **q** Stop and exit.

28895 &lt;EOF&gt; Same as q.

28896 **!command**28897 Escape to the command interpreter to execute *command*.28898 **\*** Write a usage summary for the possible responses described here.28899 **OPTIONS**

28900 The *uupick* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 28901 following option is supported:

28902 **-s** *system*28903 Process only files sent from *system*.28904 **OPERANDS**

28905 None.

28906 **STDIN**

28907 Used to read the user’s response to each file or directory prompt.

28908 **INPUT FILES**

28909 The files to be copied are regular files.

28910 **ENVIRONMENT VARIABLES**28911 The following environment variables affect the execution of *uupick*:

28912 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 28913 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 28914 default locale will be used. If any of the internationalisation variables contains an  
 28915 invalid setting, the utility will behave as if none of the variables had been defined.

28916 **LC\_ALL**

28917 If set to a non-empty string value, override the values of all the other  
 28918 internationalisation variables.

28919 *LC\_CTYPE*  
28920 Determine the locale for the interpretation of sequences of bytes of text data as  
28921 characters (for example, single- as opposed to multi-byte characters in arguments and  
28922 input files).

28923 *LC\_MESSAGES*  
28924 Determine the locale that should be used to affect the format and contents of diagnostic  
28925 messages written to standard error, and informative messages written to standard  
28926 output.

28927 *NLSPATH*  
28928 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

28929 **ASYNCHRONOUS EVENTS**  
28930 Default.

28931 **STDOUT**  
28932 Prompts are written to standard output in an unspecified format. The prompt will contain at  
28933 least the sending system name and the name of the subject file or directory.

28934 **STDERR**  
28935 Used only for diagnostic messages.

28936 **OUTPUT FILES**  
28937 The output files are copies of the input files.

28938 **EXTENDED DESCRIPTION**  
28939 None.

28940 **EXIT STATUS**  
28941 The following exit values are returned:  
28942 0 Successful completion.  
28943 >0 An error occurred.

28944 **CONSEQUENCES OF ERRORS**  
28945 Default.

28946 **APPLICATION USAGE**  
28947 There is no option (such as the *SHELL* variable) to specify a different command interpreter for  
28948 use with *!command*.

28949 Writing a file using *p* can cause problems on some terminals if the file is not a text file or  
28950 contains control characters.

28951 **EXAMPLES**  
28952 None.

28953 **FUTURE DIRECTIONS**  
28954 None.

28955 **SEE ALSO**  
28956 *uucp, uuto, uustat, uux*.



28957 **CHANGE HISTORY**

28958 First released in Issue 2.

28959 **Issue 4**

28960 Format reorganised.

28961 Utility Syntax Guidelines support mandated.

28962 Internationalised environment variable support mandated.

28963 **Issue 5**

28964 Marked LEGACY.

28965 **NAME**

28966 uustat — uucp status inquiry and job control

28967 **SYNOPSIS**28968 UN EX uustat [ -q | -k *jobid* | -r *jobid* ]28969 EX uustat [-s *system*][ -u *user* ]28970 **DESCRIPTION**28971 The *uustat* utility displays the status of, or cancels, previously specified *uucp* requests, or  
28972 provides general status on *uucp* connections to other systems.28973 When no options are given, *uustat* writes to standard output the status of all *uucp* requests  
28974 issued by the current user.28975 Typical implementations of this utility require a communications line configured to use the **XBD**  
28976 specification, **Chapter 9, General Terminal Interface**, but other communications means may be  
28977 used. On systems where there are no available communications means (either temporarily or  
28978 permanently), this utility will write an error message describing the problem and exit with a  
28979 non-zero exit status.28980 **OPTIONS**28981 The *uustat* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
28982 following options are supported:28983 UN **-q** Write the jobs queued for each machine.28984 **-k *jobid***28985 Kill the *uucp* request whose job identification is *jobid*. The killed *uucp* request must  
28986 belong to the person invoking *uustat* unless that user has appropriate privileges.28987 UN **-r *jobid*** Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification  
28988 time is set to the current time. This prevents the cleanup program from deleting the job  
28989 until the jobs modification time reaches the limit imposed by the program.28990 **-s *system***28991 Write the status of all *uucp* requests for remote system *system*.28992 **-u *user*** Write the status of all *uucp* requests issued by *user*.28993 **OPERANDS**

28994 None.

28995 **STDIN**

28996 Not used.

28997 **INPUT FILES**

28998 None.

28999 **ENVIRONMENT VARIABLES**29000 The following environment variables affect the execution of *uustat*:29001 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
29002 *LANG* is unset or null, the corresponding value from the implementation-dependent  
29003 default locale will be used. If any of the internationalisation variables contains an  
29004 invalid setting, the utility will behave as if none of the variables had been defined.29005 **LC\_ALL**29006 If set to a non-empty string value, override the values of all the other  
29007 internationalisation variables.

29008        **LC\_CTYPE**  
 29009            Determine the locale for the interpretation of sequences of bytes of text data as  
 29010            characters (for example, single- as opposed to multi-byte characters in arguments).

29011        **LC\_MESSAGES**  
 29012            Determine the locale that should be used to affect the format and contents of diagnostic  
 29013            messages written to standard error, and informative messages written to standard  
 29014            output.

29015        **LC\_TIME**  
 29016            Determine the format of date and time strings output by *uustat*.

29017        **NLSPATH**  
 29018            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

29019        **TZ**        Determine the timezone used with date and time strings.

29020 **ASYNCHRONOUS EVENTS**

29021        Default.

29022 **STDOUT**  
 29023        The standard output consists of information about each job selected, in an unspecified format.  
 29024        The information will include at least the job ID, the user ID or name and the remote system  
 29025        name.

29026 **STDERR**  
 29027        Used only for diagnostic messages.

29028 **OUTPUT FILES**  
 29029        None.

29030 **EXTENDED DESCRIPTION**  
 29031        None.

29032 **EXIT STATUS**  
 29033        The following exit values are returned:

29034        0    Successful completion.  
 29035        >0   An error occurred.

29036 **CONSEQUENCES OF ERRORS**  
 29037        Default.

29038 **APPLICATION USAGE**  
 29039        None.

29040 **EXAMPLES**  
 29041        None.

29042 **FUTURE DIRECTIONS**  
 29043        None.

29044 **SEE ALSO**  
 29045        *uucp*.

29046 **CHANGE HISTORY**

29047 First released in Issue 2.

29048 **Issue 4**

29049 Format reorganised.

29050 Utility Syntax Guidelines support mandated.

29051 Internationalised environment variable support mandated.

29052 Presence of the utility mandated, even on systems where no communications are available.

29053 **NAME**

29054           uuto — send public system-to-system file copies (**LEGACY**)

29055 **SYNOPSIS**

29056 UN EX   uuto [-mp] *source-file... destination*

29057 **DESCRIPTION**

29058           The *uuto* utility sends *source-files* to *destination*. The *uuto* utility uses the *uucp* facility to send  
 29059           files, while it allows the local system to control the file access. A *source-file* name is a pathname  
 29060           on the user's machine.

29061           The files (or subtrees if directories are specified) are sent to a public directory (called “*PUBDIR*”;  
 29062           the actual location of *PUBDIR* is implementation-dependent) on *system*. Specifically, the files are  
 29063           sent to the directory:

29064                 *PUBDIR/receive/user/fsystem*

29065           where *user* is the recipient, and *fsystem* is the sending system.

29066           The recipient is notified by mail of the arrival of files.

29067           Typical implementations of this utility require a communications line configured to use the **XBD**  
 29068           specification, **Chapter 9, General Terminal Interface**, but other communications means may be  
 29069           used. On systems where there are no available communications means (either temporarily or  
 29070           permanently), this utility will write an error message describing the problem and exit with a  
 29071           non-zero exit status.

29072           The *uuto* utility cannot guarantee support for all character encodings in all circumstances. For  
 29073           example, transmission data may be restricted to 7-bits by the underlying network, 8-bit data and  
 29074           filenames need not be portable to non-internationalised systems, and so on. Under these  
 29075           circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
 29076           standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used  
 29077           and that only characters defined in the Portable Filename Character Set be used for naming files.

29078 **OPTIONS**

29079           The *uuto* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 29080           following options are supported:

29081           **-m**       Send mail to the sender when the copy is complete.

29082           **-p**       Copy the source file into the spool directory before transmission.

29083 **OPERANDS**

29084           The following operands are supported:

29085           *destination*

29086                 A string of the form:

29087                 *system-name!user*

29088                 where *system-name* is taken from a list of system names that *uucp* knows about; see  
 29089                 *uuname*. The argument *user* is the login name of someone on the specified system. The  
 29090                 destination *system-name* can also be a list of names such as:

29091                 *system-name!system-name!...!system-name!user*

29092                 in which case, an attempt is made to send the file via the specified route to the  
 29093                 destination. Care should be taken to ensure that intermediate nodes in the route are  
 29094                 willing to forward information.

29095 *source-file*  
 29096 A pathname of a file on the local system to be copied to *destination*.

29097 **STDIN**  
 29098 Not used.

29099 **INPUT FILES**  
 29100 The files to be copied are regular files.

29101 **ENVIRONMENT VARIABLES**  
 29102 The following environment variables affect the execution of *uuto*:

29103 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 29104 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 29105 default locale will be used. If any of the internationalisation variables contains an  
 29106 invalid setting, the utility will behave as if none of the variables had been defined.

29107 *LC\_ALL*  
 29108 If set to a non-empty string value, override the values of all the other  
 29109 internationalisation variables.

29110 *LC\_CTYPE*  
 29111 Determine the locale for the interpretation of sequences of bytes of text data as  
 29112 characters (for example, single- as opposed to multi-byte characters in arguments and  
 29113 input files).

29114 *LC\_MESSAGES*  
 29115 Determine the locale that should be used to affect the format and contents of diagnostic  
 29116 messages written to standard error, and informative messages written to standard  
 29117 output.

29118 *NLSPATH*  
 29119 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

29120 **ASYNCHRONOUS EVENTS**  
 29121 Default.

29122 **STDOUT**  
 29123 None.

29124 **STDERR**  
 29125 Used only for diagnostic messages.

29126 **OUTPUT FILES**  
 29127 The output files (which may be on other systems) are copies of the input files.

29128 **EXTENDED DESCRIPTION**  
 29129 None.

29130 **EXIT STATUS**  
 29131 The following exit values are returned:  
 29132 0 Successful completion.  
 29133 >0 An error occurred.

29134 **CONSEQUENCES OF ERRORS**  
 29135 Default.

29136 **APPLICATION USAGE**  
 29137 None.

29138 **EXAMPLES**

29139 None.

29140 **FUTURE DIRECTIONS**

29141 None.

29142 **SEE ALSO**29143 *mailx, uucp, uuencode, uupick, uustat, uux.*29144 **CHANGE HISTORY**

29145 First released in Issue 2. |

29146 **Issue 4**

29147 Format reorganised.

29148 Utility Syntax Guidelines support mandated.

29149 Internationalised environment variable support mandated.

29150 Presence of the utility mandated, even on systems where no communications are available. |

29151 **Issue 5** |

29152 Marked LEGACY.

## 29153 NAME

29154 uux — remote command execution

## 29155 SYNOPSIS

29156 EX uux [-np] *command-string*29157 UN EX uux [-jnp] *command-string*29158 UN OB EXuux [-][ -jn] *command-string*

## 29159 DESCRIPTION

29160 The *uux* utility will gather zero or more files from various systems, execute a shell pipeline (see  
 29161 Section 2.9 on page 45) on a specified system, and then send the standard output of the  
 29162 command to a file on a specified system. Only the first command of a pipeline can have a  
 29163 “*system-name!*” prefix. All other commands in the pipeline are executed on the system of the  
 29164 first command.

29165 The following restrictions are applicable to the shell pipeline processed by *uux*:

- 29166 • In gathering files from different systems, pathname expansion is not performed by *uux*.
- 29167 Thus, a request such as:

29168 uux "c89 remsys!~/\*.c"

29169 would attempt to copy the file named literally \*.c to the local system.

- 29170 • The redirection operators >>, <<, >| and >& cannot be used.
- 29171 • The reserved word ! cannot be used at the head of the pipeline to modify the exit status.
- 29172 • Alias substitution is not performed.

29173 A filename can be specified as for *uucp*; it can be an absolute pathname, a pathname preceded by  
 29174 ~*name* (which is replaced by the corresponding login directory), a pathname specified as ~/*dest*  
 29175 (*dest* is prefixed by the public directory called “*PUBDIR*”; the actual location of *PUBDIR* is  
 29176 implementation-dependent), or a simple filename (which is prefixed by *uux* with the current  
 29177 directory). See *uucp* for the details.

29178 The execution of commands on remote systems takes place in an execution directory known to  
 29179 the *uucp* system. All files required for the execution will be put into this directory unless they  
 29180 already reside on that machine. Therefore, the non-local filenames (without path or machine  
 29181 reference) must be unique within the *uux* request.

29182 The *uux* utility will attempt to get all files to the execution system. For files that are output files,  
 29183 the filename must be escaped using parentheses.

29184 The remote system will notify the user by mail if the requested command on the remote system  
 29185 was disallowed or the files were not accessible. This notification can be turned off by the *-n*  
 29186 option.

29187 Typical implementations of this utility require a communications line configured to use the **XBD**  
 29188 specification, **Chapter 9, General Terminal Interface**, but other communications means may be  
 29189 used. On systems where there are no available communications means (either temporarily or  
 29190 permanently), this utility will write an error message describing the problem and exit with a  
 29191 non-zero exit status.

29192 The *uux* utility cannot guarantee support for all character encodings in all circumstances. For  
 29193 example, transmission data may be restricted to 7-bits by the underlying network, 8-bit data and  
 29194 filenames need not be portable to non-internationalised systems, and so on. Under these  
 29195 circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
 29196 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used



29197 and that only characters defined in the Portable Filename Character Set be used for naming files.

## 29198 OPTIONS

29199 The *uux* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 29200 that **–** is supported as an option in the obsolescent version, rather than an operand. The  
 29201 following options are supported:

29202 **–p**

29203 OB **–** Make the standard input to *uux* the standard input to the *command-string*.

29204 UN **–j** Write the job identification string to standard output. This job identification can be  
 29205 used by *uustat* to obtain the status or terminate a job.

29206 **–n** Do not notify the user if the command fails.

## 29207 OPERANDS

29208 The following operands are supported:

29209 *command-string*

29210 A string made up of one or more arguments that are similar to normal command  
 29211 arguments, except that the command and any filenames can be prefixed by “*system-*  
 29212 *name!*”. A null *system-name* is interpreted as the local system.

## 29213 STDIN

29214 The standard input is not used unless the **–** or **–p** option is specified; in those cases, the standard  
 29215 input is made the standard input of the *command-string*.

## 29216 INPUT FILES

29217 Input files are selected according to the contents of *command-string*.

## 29218 ENVIRONMENT VARIABLES

29219 The following environment variables affect the execution of *uux*:

29220 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 29221 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 29222 default locale will be used. If any of the internationalisation variables contains an  
 29223 invalid setting, the utility will behave as if none of the variables had been defined.

29224 **LC\_ALL**

29225 If set to a non-empty string value, override the values of all the other  
 29226 internationalisation variables.

29227 **LC\_CTYPE**

29228 Determine the locale for the interpretation of sequences of bytes of text data as  
 29229 characters (for example, single- as opposed to multi-byte characters in arguments).

29230 **LC\_MESSAGES**

29231 Determine the locale that should be used to affect the format and contents of diagnostic  
 29232 messages written to standard error.

29233 **NLSPATH**

29234 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

## 29235 ASYNCHRONOUS EVENTS

29236 Default.

## 29237 STDOUT

29238 The standard output is not used unless the **–j** option is specified; in that case, the job  
 29239 identification string is written to standard output in the following format:

29240 "%s\n", <jobid>

## 29241 **STDERR**

29242 Used only for diagnostic messages.

## 29243 **OUTPUT FILES**

29244 Output files are created or written, or both, according to the contents of *command-string*.

29245 If the **-n** is not used, mail files will be modified following any command or file-access failures on  
29246 the remote system.

## 29247 **EXTENDED DESCRIPTION**

29248 None.

## 29249 **EXIT STATUS**

29250 The following exit values are returned:

29251 0 Successful completion.

29252 >0 An error occurred.

## 29253 **CONSEQUENCES OF ERRORS**

29254 Default.

## 29255 **APPLICATION USAGE**

29256 Note that, for security reasons, many installations will limit the list of commands executable on  
29257 behalf of an incoming request from *uux*. Many sites will permit little more than the receipt of  
29258 mail via *uux*.

29259 Any characters special to the command interpreter should be quoted either by quoting the entire  
29260 *command-string* or quoting the special characters as individual arguments.

29261 As noted in *uucp*, shell pattern matching notation characters appearing in pathnames are  
29262 expanded on the appropriate local system. This is done under the control of local settings of  
29263 *LC\_COLLATE* and *LC\_CTYPE*. Thus, care should be taken when using bracketed filename  
29264 patterns, as collation and typing rules may vary from one system to another. Also be aware that  
29265 certain types of expression (that is, equivalence classes, character classes and collating symbols)  
29266 need not be supported on non-internationalised systems.

## 29267 **EXAMPLES**

29268 1. The following command gets *file1* from system **a** and *file2* file from system **b**, executes *diff*  
29269 on the local system, and puts the results in *file.diff* in the local *PUBDIR* directory.  
29270 (*PUBDIR* is the *uucp* public directory on the local system.)

29271 `uux "!diff a!/usr/file1 b!/a4/file2 >!/~/file.diff"`

29272 2. The following command will fail because *uux* places all files copied to a system in the same  
29273 working directory. Although the files **xyz** are from two different systems, their filenames  
29274 are the same and will conflict.

29275 `uux "!diff a!/usr1/xyz b!/usr2/xyz >!/~/xyz.diff"`

29276 3. The following command will succeed (assuming *diff* is permitted on system **a**) because the  
29277 file local to system **a** is not copied to the working directory, and hence does not conflict the  
29278 file from system **c**.

29279 `uux "a!diff a!/usr/xyz c!/usr/xyz >!/~/xyz.diff"`

## 29280 **FUTURE DIRECTIONS**

29281 A version of *uux* that fully supports the **XBD** specification, **Section 10.2, Utility Syntax**  
29282 **Guidelines** may be introduced in a future issue.

29283 **SEE ALSO**29284 *uucp, uuencode, uustat.*29285 **CHANGE HISTORY**

29286 First released in Issue 2. |

29287 **Issue 4**

29288 Format reorganised.

29289 Exceptions to Utility Syntax Guidelines conformance noted. |

29290 Internationalised environment variable support mandated.

29291 Presence of the utility mandated, even on systems where no communications are available.

## 29292 NAME

29293 val — validate SCCS files (**DEVELOPMENT**)

## 29294 SYNOPSIS

29295 EX val -

29296 val [-s][-m *name*][-r *SID*][-y *type*] *file...*

## 29297 DESCRIPTION

29298 The *val* utility determines if the specified *file* is an SCCS file meeting the characteristics specified  
 29299 by the options.

## 29300 OPTIONS

29301 The *val* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 29302 that the usage of the - operand is not strictly as intended by the guidelines (that is, reading  
 29303 options and operands from standard input). The following options are supported:

29304 -m *name*29305 Specify a *name*, which is compared with the SCCS %M% keyword in *file*. (See *get*).

29306 -r *SID* Specify a *SID* (SCCS Identification String), an SCCS delta number. A check is made to  
 29307 determine if the *SID* is ambiguous (for example, -r 1 is ambiguous because it  
 29308 physically does not exist but implies 1.1, 1.2, and so on, which may exist) or invalid (for  
 29309 example, -r 1.0 or -r 1.1.0 are invalid because neither case can exist as a valid delta  
 29310 number). If the *SID* is valid and not ambiguous, a check is made to determine if it  
 29311 actually exists.

29312 -s Silence the diagnostic message normally written to standard output for any error that is  
 29313 detected while processing each named file on a given command line.

29314 -y *type* Specify a *type*, which is compared with the SCCS %Y% keyword in *file*. (See *get*).

## 29315 OPERANDS

29316 The following operands are supported:

29317 *file* A pathname of an existing SCCS file. If a single instance *file* is specified as -, and if no  
 29318 options are specified, the standard input is read: each line is independently processed  
 29319 as if it were a command-line argument list. (However, the line is not subjected to any  
 29320 of the shell word expansions, such as parameter expansion or quote removal.)

## 29321 STDIN

29322 The standard input is a text file used only when the *file* operand is specified as -.

## 29323 INPUT FILES

29324 Any SCCS files processed are files of an unspecified format.

## 29325 ENVIRONMENT VARIABLES

29326 The following environment variables affect the execution of *val*:

29327 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 29328 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 29329 default locale will be used. If any of the internationalisation variables contains an  
 29330 invalid setting, the utility will behave as if none of the variables had been defined.

29331 *LC\_ALL*

29332 If set to a non-empty string value, override the values of all the other  
 29333 internationalisation variables.

29334        **LC\_CTYPE**  
 29335            Determine the locale for the interpretation of sequences of bytes of text data as  
 29336            characters (for example, single- as opposed to multi-byte characters in arguments and  
 29337            input files).

29338        **LC\_MESSAGES**  
 29339            Determine the locale that should be used to affect the format and contents of diagnostic  
 29340            messages written to standard error. and informative messages written to standard  
 29341            output.

29342        **NLSPATH**  
 29343            Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

29344        **ASYNCHRONOUS EVENTS**  
 29345            Default.

29346        **STDOUT**  
 29347            The standard output consists of informative messages about either: (1) each file processed, or;  
 29348            (2) each command line read from standard input.

29349            If the standard input is not used, for each *file* operand yielding a discrepancy, the output line has  
 29350            the following format:

29351                " %s: %s\n", <pathname>, <unspecified string>

29352            If standard input is used, a line of input is written before each of the preceding lines for files  
 29353            containing discrepancies:

29354                "%s:\n", <input line>

29355        **STDERR**  
 29356            Not used.

29357        **OUTPUT FILES**  
 29358            None.

29359        **EXTENDED DESCRIPTION**  
 29360            None.

29361        **EXIT STATUS**  
 29362            The 8-bit code returned by *val* is a disjunction of the possible errors, that is, it can be interpreted  
 29363            as a bit string where set bits are interpreted as follows:

29364            0x80    =    Missing file argument.  
 29365            0x40    =    Unknown or duplicate option.  
 29366            0x20    =    Corrupted SCCS file.  
 29367            0x10    =    Cannot open file or file not SCCS.  
 29368            0x08    =    *SID* is invalid or ambiguous.  
 29369            0x04    =    *SID* does not exist.  
 29370            0x02    =    %Y1%, -y mismatch.  
 29371            0x01    =    %M %, -m mismatch.

29372            Note that *val* can process two or more files on a given command line and can process multiple  
 29373            command lines (when reading the standard input). In these cases an aggregate code is returned:  
 29374            a logical OR of the codes generated for each command line and file processed.

29375        **CONSEQUENCES OF ERRORS**  
 29376            Default.

29377 **APPLICATION USAGE**

29378           Since the *val* exit status sets the 0x80 bit, shell applications checking \$? cannot tell if it  
 29379           terminated due to a missing file argument or receipt of a signal.

29380 **EXAMPLES**

29381           In a directory with three SCCS files, **s.x** (of **t** type “text”), **s.y** and **s.z** (a corrupted file), the  
 29382           following command could produce the output shown:

```
29383 val - <<EOF
29384 -y source s.x
29385 -m y s.y
29386 s.z
29387 EOF

29388 -y source s.x

29389 s.x: %Y%, -y mismatch
29390 s.z

29391 s.z: corrupted SCCS file
```

29392 **FUTURE DIRECTIONS**

29393           None.

29394 **SEE ALSO**

29395           *admin, delta, get, prs.*

29396 **CHANGE HISTORY**

29397           First released in Issue 2.

29398 **Issue 4**

29399           Format reorganised.

29400           Exceptions to Utility Syntax Guidelines conformance noted.

29401           Internationalised environment variable support mandated.

## 29402 NAME

29403 vi — screen-oriented (visual) display editor

## 29404 SYNOPSIS

29405 EX vi [-rR][-l][-c *command*][-t *tagstring*][-w *size*][*file...*]29406 EX OB vi [-rR][-l][+*command*][-t *tagstring*][-w *size*][*file...*]

## 29407 DESCRIPTION

29408 The *vi* (visual) utility is a screen-oriented text editor. The user can switch back and forth  
 29409 between *vi* and the line editor *ex* and execute *ex* commands from within *vi*.

29410 When using *vi*, the terminal screen acts as a window into the editing buffer. Changes made to  
 29411 the editing buffer are reflected in the screen display; the position of the cursor on the screen  
 29412 indicates the position within the editing buffer.

29413 Certain block-mode terminals do not have all the capabilities necessary to support the complete  
 29414 *vi* definition. When these commands cannot be supported on such terminals, this condition will  
 29415 not produce an error message such as “not an editor command” or report a syntax error. The  
 29416 implementation may either accept the commands and produce results on the screen that are the  
 29417 result of an unsuccessful attempt to meet the requirements of this specification or report an error  
 29418 describing the terminal-related deficiency.

## 29419 OPTIONS

29420 OB The *vi* utility supports the XBD specification, **Section 10.2, Utility Syntax Guidelines** except for  
 29421 the obsolescent *+command* option.

29422 The following options are supported:

29423 -c *command*29424 OB +*command*

29425 Begin editing by executing the specified *ex* command-mode commands. As with  
 29426 normal *ex* command-line entries, the *command* option-argument can consist of multiple  
 29427 *ex* commands separated by vertical-line characters (|). The use of commands that enter  
 29428 input mode in this manner produces undefined results.

29429 EX -l Set lisp mode (see **Edit Options in ex** on page 325).

29430 -r Attempt to recover the named *files* after an editor or system crash, after the editor has  
 29431 been terminated by a signal or after the use of a **pre** (*ex*) editor command.

29432 If no *file* operands are given, all other options, the *EXINIT* variable, and any *.exrc* files  
 29433 will be ignored; a list of all recoverable files available to the invoking user will be  
 29434 written; and *vi* will exit without reading files or processing user commands.

29435 -R Set read-only mode, preventing accidental overwriting of the files. Any command that  
 29436 would write to a file requires the “!” suffix (see, for example, the **write** command) to be  
 29437 effective in this mode.

29438 -t *tagstring*

29439 Edit the file containing the specified *tagstring* and set the initial position within the edit  
 29440 buffer to the point of definition of the tag. (See *ctags*.) The tags feature represented by  
 29441 -t *tagstring* and the **ta** command is optional. It is provided on any system that also  
 29442 provides a conforming implementation of *ctags*; otherwise, the use of -t produces  
 29443 undefined results.

29444            **-w** *size* Set the value of the *window* editor option to *size*. See the **window** option.

29445 OB        If both the **-t** *tagstring* and **-c** *command* (or the obsolescent *+command*) options are given, the

29446            **-t** *tagstring* will be processed first; that is, the file containing the tag is selected by **-t** and then

29447            the command is executed.

## 29448 OPERANDS

29449            The following operand is supported:

29450            *file*        A pathname of a file to be edited.

## 29451 STDIN

29452            The standard input consists of a series of commands and input text, as described in the

29453            EXTENDED DESCRIPTION section.

## 29454 INPUT FILES

29455            Input files must be text files or files that would be text files except for an incomplete last line that

29456            is not longer than {LINE\_MAX} – 1 bytes in length and contains no NUL characters. The editing

29457            of other forms of files may optionally be allowed by *vi* implementations.

29458            The **.exrc** files (see the EXTENDED DESCRIPTION section) must be text files consisting of *ex*

29459            commands.

29460            By default, *vi* will read lines from the files to be edited without interpreting any of those lines as

29461            any form of editor command.

## 29462 ENVIRONMENT VARIABLES

29463            The following environment variables affect the execution of *vi*:

### 29464 COLUMNS

29465            Override the system-selected horizontal screen size. See the **XBD** specification,

29466            **Chapter 6, Environment Variables** for valid values and results when it is unset or null.

29467            **EXINIT** Determine a list of *ex* commands that will be executed on editor startup, before reading

29468            the first file. The list can contain multiple commands by separating them using a

29469            vertical-line (|) character. See also the EXTENDED DESCRIPTION section for more

29470            details of the initialisation phase.

29471            **HOME** Determine a pathname of a directory that will be searched for an editor startup file

29472            named **.exrc**; see the EXTENDED DESCRIPTION section.

29473            **LANG** Provide a default value for the internationalisation variables that are unset or null. If

29474            **LANG** is unset or null, the corresponding value from the implementation-dependent

29475            default locale will be used. If any of the internationalisation variables contains an

29476            invalid setting, the utility will behave as if none of the variables had been defined.

### 29477 LC\_ALL

29478            If set to a non-empty string value, override the values of all the other

29479            internationalisation variables.

### 29480 LC\_COLLATE

29481            Determine the locale for the behaviour of ranges, equivalence classes and multi-

29482            character collating elements within regular expressions.

### 29483 LC\_CTYPE

29484            Determine the locale for the interpretation of sequences of bytes of text data as

29485            characters (for example, single- as opposed to multi-byte characters in arguments and

29486            input files), the behaviour of character classes within regular expressions, the

29487            classification of characters as upper- or lower-case letters, the case conversion of letters,

29488            and the detection of word boundaries.



|       |                             |                                                                                                                       |
|-------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------|
| 29489 | <b>LC_MESSAGES</b>          |                                                                                                                       |
| 29490 |                             | Determine the locale that should be used to affect the format and contents of diagnostic                              |
| 29491 |                             | messages written to standard error and informative messages written to standard                                       |
| 29492 |                             | output.                                                                                                               |
| 29493 | <b>NLSPATH</b>              |                                                                                                                       |
| 29494 |                             | Determine the location of message catalogues for the processing of <i>LC_MESSAGES</i> .                               |
| 29495 | <b>LINES</b>                | Override the system-selected vertical screen size, used as the number of lines in a                                   |
| 29496 |                             | screenful and the vertical screen size in visual mode. See the <b>XBD</b> specification,                              |
| 29497 |                             | <b>Chapter 6, Environment Variables</b> for valid values and results when it is unset or null.                        |
| 29498 | <b>PATH</b>                 | Determine the search path for the shell command specified in the editor commands                                      |
| 29499 |                             | <b>shell</b> , <b>read</b> and <b>write</b> and the visual-mode command "!"; see the description of                   |
| 29500 |                             | command search and execution in <b>Command Search and Execution</b> on page 47.                                       |
| 29501 | <b>SHELL</b>                | Determine the preferred command-line interpreter for use in "!", <b>shell</b> , <b>read</b> and other                 |
| 29502 |                             | commands with an operand of the form <i>!string</i> . For the <b>shell</b> command the program                        |
| 29503 |                             | will be invoked with the single argument <i>-i</i> , for all others it will be invoked with the                       |
| 29504 |                             | two arguments <i>-c</i> and <i>string</i> . If this variable is null or not set, the <i>sh</i> utility will be        |
| 29505 |                             | used.                                                                                                                 |
| 29506 | <b>TERM</b>                 | Determine the name of the terminal type. If this variable is unset or null, an                                        |
| 29507 |                             | unspecified default terminal type will be used.                                                                       |
| 29508 | <b>ASYNCHRONOUS EVENTS</b>  |                                                                                                                       |
| 29509 |                             | The following actions will be taken upon receipt of signals:                                                          |
| 29510 | <b>SIGINT</b>               | The current editor command will be aborted. The editor will prompt for another                                        |
| 29511 |                             | command.                                                                                                              |
| 29512 | <b>SIGCONT</b>              |                                                                                                                       |
| 29513 |                             | The screen will be refreshed (if in <i>visual</i> mode).                                                              |
| 29514 | <b>SIGHUP</b>               |                                                                                                                       |
| 29515 |                             | If the current buffer has changed since the last <i>e</i> or <i>w</i> command, <i>vi</i> will attempt to save         |
| 29516 |                             | the current file in a state such that it can be recovered later by an <i>ex</i> or <i>vi -r</i> command.              |
| 29517 |                             | The action taken for all other signals is unspecified.                                                                |
| 29518 | <b>STDOUT</b>               |                                                                                                                       |
| 29519 |                             | If standard output is a terminal device, it may be used for writing prompts to the user, for                          |
| 29520 |                             | informational messages and for writing lines from the file. If standard output is not a terminal                      |
| 29521 |                             | device, undefined results occur.                                                                                      |
| 29522 | <b>STDERR</b>               |                                                                                                                       |
| 29523 |                             | Used only for diagnostic messages.                                                                                    |
| 29524 | <b>OUTPUT FILES</b>         |                                                                                                                       |
| 29525 |                             | The output from <i>vi</i> are text files that are identical to the input files if no changes have been made           |
| 29526 |                             | to those files by commands, with the exception that in all cases where a forced session                               |
| 29527 |                             | termination (the <i>ex</i> command <i>q!</i> ) has not been issued prior to any write of the file, a trailing         |
| 29528 |                             | newline character will be added to the last line of the file if one was not present in the input.                     |
| 29529 | <b>EXTENDED DESCRIPTION</b> |                                                                                                                       |
| 29530 |                             | Only the visual mode of the editor is described in this section. See the <i>ex</i> utility for additional             |
| 29531 |                             | editing capabilities used in <i>vi</i> .                                                                              |
| 29532 |                             | The pathname of the file being edited by <i>vi</i> is the <i>current</i> file. The text of the file will be read into |
| 29533 |                             | a <i>buffer</i> , and all editing changes will be performed in this buffer; changes will have no effect on            |

the file until the buffer is written out explicitly. Lines in the buffer may each be limited to {LINE\_MAX} bytes and an error message may be displayed if the limit is exceeded during editing.

The *alternative* pathname is the name of the last file mentioned in an editor command, or the previous current pathname if the last file mentioned became the current file. When the character "%" appears in a pathname entered as part of a command argument, the character will be replaced by the current pathname; the character "#" will be replaced by the alternative pathname. The characters "%" and "#" can be escaped by preceding them with a backslash.

During initialisation, before the first file is read or any user commands from the terminal are processed, if the environment variable *EXINIT* is set, the editor will execute the *ex* commands contained in that variable. If the variable is not set, *vi* will attempt to read *ex* commands from the file *\$HOME/.exrc* (the file *.exrc* in the directory referred to by the *HOME* environment variable). If and only if *EXINIT* or *\$HOME/.exrc* sets the editor option *exrc*, *vi* finally will attempt to read *ex* commands from a file *.exrc* in the current directory. In the event that *EXINIT* is not set and the current directory is the user's home directory, any *.exrc* file will only be processed once. No *.exrc* file will be read unless it is owned by the same user ID as the effective user ID of the process. After any *.exrc* files are processed, any commands specified by the *-c* option will be processed.

After initialisation, *vi* will be in *command* mode; *input* mode can be entered by several commands used to insert or change text. In input mode, an escape character can be used to return to command mode; other uses of the escape character are described in **Terminate Command or Input Mode** on page 809. The results of entering newline characters in input mode are affected by the setting of the **autoindent** editor variable.

The last (bottom) line of the screen is used to display the input for search commands ("/" and "?"), for *ex* commands (":"), and system commands ("!"). It is also used to report errors or display informational messages. The editor prompts for input for commands by displaying the command ("/", "?", ":" or "!") at the beginning of the last line of the screen. All subsequent characters will be then taken as input until a line terminator is entered.

An interrupt typed during text input, or during the input of a command on the bottom line, will terminate the input (or cancel the command) and will return the editor to command mode. During command mode, an interrupt will alert the terminal. In general, an alert indicates an error (such as unrecognised key).

Lines displayed on the screen containing only a tilde (~) indicate that the last line above them is the last line in the editing buffer (the ~ lines are past the end of the editing buffer).

There may be lines on the screen marked with an "@". These indicate space on the screen not corresponding to lines in the file. This will happen if there is space at the bottom of the screen for some text, but the next line of the buffer will not fit. (It may also happen on a terminal with limited local intelligence. In this case, these lines can be removed by entering a <control>-R, forcing the editor to refresh the screen.)

## Command Descriptions in vi

The cursor, in general, is placed on the current line and in the current column position as noted for each command described below. If the current line is not in the display window, then the display window will be either scrolled or refreshed to cause the current line to be in the display window. If the screen is refreshed, the current line will be positioned as close to the center of the display window as possible. If the current line is less than one-half window lines from the beginning of the editing buffer, the first line of the buffer will be displayed on the first line of the display window. If the current line is less than one-half window lines from the end of the

29581 editing buffer, the remaining lines of the display window after the last line of the file will contain  
 29582 only a ~ character in column position 1. If the screen is scrolled rather than refreshed, the current  
 29583 line will be placed at the top of the display window if the current line is before the first line  
 29584 displayed. If the current line is after the last line displayed, the display window will be scrolled  
 29585 up and the current line will be placed on the last line of the display window.

29586 As stated previously, the cursor is generally placed on the current column position of the current  
 29587 line. The one exception to this is if the current column position is beyond the end of the current  
 29588 line. In this case, the cursor will be placed on the last character of the current line. However, the  
 29589 current column position value will not be altered by this. Thus, if the current line changes to a  
 29590 longer line, the cursor will be moved back out to the current column position or to the end of the  
 29591 line if it is shorter than the current column.

29592 Unless otherwise specified, the commands are interpreted in command mode and have no  
 29593 special effect in input mode.

29594 Some of the following command descriptions (such as a, A, c, and so on) move the cursor and  
 29595 enter input mode. The indications of *Current line* and *Current column* are for the cursor  
 29596 movement up to the beginning of input mode. Depending on the text inserted, the cursor will  
 29597 generally wind up in a totally different location by the time command mode is reentered. The  
 29598 cursor movement that accompanies the transition from input mode to command mode is  
 29599 described in **Terminate Command or Input Mode** on page 809.

29600 The following symbols are used in this section to represent arguments to commands, to describe  
 29601 commands or to specify the new values of the current line and column indicators following  
 29602 execution of the command.

29603 **bigword** A maximal sequence of non-blank characters preceded and followed by blank  
 29604 characters or the beginning or end of a line or the file.

29605 **buffer** One of a number of named areas for saving text. Commands that change or delete text  
 29606 can be preceded by a buffer specification argument *buffer*. It is specified in the form "c,  
 29607 where c represents the name of a buffer, one of the lower-case letters of the POSIX  
 29608 locale. Specifying *buffer* will cause the area of text affected by the command to be  
 29609 stored into the buffer as it was before the command took effect. This argument is also  
 29610 used on the put commands (p and P) to specify the buffer that will provide the text to  
 29611 insert. When a command synopsis shows both [ *buffer* ] and [ *count* ] preceding the  
 29612 command letter, either can precede the other.

29613 If the buffer name is specified in upper-case, and the buffer is to be modified (as with a  
 29614 deletion or yanking command) the buffer will be appended to rather than being  
 29615 overwritten. If the buffer is not to be modified (as in a visual mode put commands) the  
 29616 buffer name can be specified in lower-case or upper-case with the same results. There  
 29617 will be also one unnamed buffer, which is the repository for all text deleted (with the  
 29618 **delete** or visual mode d command) or yanked (with the **yank** or visual mode y  
 29619 command) when no buffer is specified.

29620 There are also numbered buffers, 1 to 9, inclusive, which are accessible only from visual  
 29621 mode. These buffers are special in that, in visual mode, when deleted text is placed in  
 29622 the unnamed buffer, it also will be placed in buffer 1, the previous contents of buffer 1  
 29623 will be placed in buffer 2, and so on. Any text in buffer 9 will be lost. Text that is  
 29624 yanked (or otherwise copied) into the unnamed buffer will not modify the numbered  
 29625 buffers. Text cannot be placed directly into the numbered buffers although it can be  
 29626 retrieved from them by using a visual mode put command with the buffer name given  
 29627 as a number. When the *buffer* modifier is not used in the commands below, the  
 29628 unnamed buffer is the default.

|       |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
|-------|---------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-------------|
| 29629 | <i>column</i>                                                                                           | The (previous) value of the current column indicator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |             |             |
| 29630 | <i>count</i>                                                                                            | A positive integer used as an optional argument to most commands, either to give a size or a position (for display or movement commands), or as a repeat count (for commands that change text). This argument is optional and defaults to 1 unless otherwise noted in the individual command description.                                                                                                                                                                                                                                                                                                                                                                                                  |             |             |
| 29631 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29632 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29633 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29634 | <i>end-of-line</i>                                                                                      | A description of the current column indicator value meaning that the current column indicator will be set always to indicate the last character of the current line. Until the current column indicator is changed, the cursor will be always positioned to the last character of whatever line is current.                                                                                                                                                                                                                                                                                                                                                                                                |             |             |
| 29635 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29636 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29637 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29638 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29639 | <i>line</i>                                                                                             | The (previous) value of the current line indicator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |             |             |
| 29640 | <i>non-blank</i>                                                                                        | A description of the current column indicator value meaning that the current column indicator will be set to the position of the first non-blank character of the current line. If the current line has no non-blank characters, the indicator will be set to the position of the last character of the line.                                                                                                                                                                                                                                                                                                                                                                                              |             |             |
| 29641 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29642 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29643 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29644 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29645 | <i>motion</i>                                                                                           | A command used as an optional trailing argument to some commands, indicating the extent of text to be affected by the command. The <i>motion</i> argument can be either the command character repeated or a cursor movement command. In the former case, exactly the current line is affected. In the latter case, the region specified will be from the current cursor position to just before the cursor position indicated by the motion command. If the command operates on lines only, then all the lines that fall partly or wholly within this region are affected. Otherwise, the exact region as described above is affected. The following commands are considered to be cursor motion commands: |             |             |
| 29646 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29647 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29648 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29649 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29650 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29651 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29652 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29653 | \$                                                                                                      | ^                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | e           | M           |
| 29654 | %                                                                                                       | “                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <control>-F | <control>-N |
| 29655 | ”                                                                                                       | <i>letter</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | F           | N           |
| 29656 | <i>letter</i>                                                                                           | {                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | f           | n           |
| 29657 | (                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | G           | <control>-P |
| 29658 | )                                                                                                       | }                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <control>-H | <space>     |
| 29659 | ,                                                                                                       | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | H           | <control>-T |
| 29660 | -                                                                                                       | <control>-B                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <control>-J | T           |
| 29661 | /                                                                                                       | B                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | j           | t           |
| 29662 | ;                                                                                                       | b                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | k           | <control>-U |
| 29663 | ?                                                                                                       | <control>-D                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | L           | W           |
| 29664 | [[                                                                                                      | <control>-E                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | l           | w           |
| 29665 | ]]                                                                                                      | E                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <control>-M | <control>-Y |
| 29666 | The optional <i>count</i> prefix available for some of the motion commands can be included              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29667 | and is considered part of the <i>motion</i> argument. For example, in <b>c2w</b> , the <b>2w</b> is the |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29668 | <i>motion</i> argument.                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29669 | <i>previous context</i>                                                                                 | A state set whenever a non-relative motion command is executed in <i>vi</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |             |             |
| 29670 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29671 | <i>paragraph</i>                                                                                        | An area of text that begins with either the beginning of a file, an empty line, or section boundary and continues until either an empty line, section boundary or the end of the editing buffer. Additional paragraph boundaries can be defined by the <b>ex paragraph</b> option.                                                                                                                                                                                                                                                                                                                                                                                                                         |             |             |
| 29672 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29673 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29674 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |
| 29675 |                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |             |

|       |                 |                                                                                                                                                                                                                                                                                                                                                              |
|-------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29676 | <i>section</i>  | An area of text that starts with a line whose first character is either a form-feed character or an open brace ({} and continues until the next section or the end of the editing buffer. The beginning and end of the editing buffer are also considered section boundaries. Additional section boundaries can be defined by the <b>ex sections</b> option. |
| 29677 |                 |                                                                                                                                                                                                                                                                                                                                                              |
| 29678 |                 |                                                                                                                                                                                                                                                                                                                                                              |
| 29679 |                 |                                                                                                                                                                                                                                                                                                                                                              |
| 29680 | <i>sentence</i> |                                                                                                                                                                                                                                                                                                                                                              |
| 29681 |                 | An area of text that begins with either the beginning of the editing buffer or the first non-blank character following the previous sentence, paragraph or section boundary                                                                                                                                                                                  |
| 29682 |                 | and continues until the end of the editing buffer or a period, exclamation point or                                                                                                                                                                                                                                                                          |
| 29683 |                 | question mark (".", "!", "?") character followed by either an end-of-line or two space                                                                                                                                                                                                                                                                       |
| 29684 |                 | characters. Any number of closing parentheses, brackets or double-quote ("), "]", "\"")                                                                                                                                                                                                                                                                      |
| 29685 |                 | characters can appear between the period, exclamation point or question mark and the                                                                                                                                                                                                                                                                         |
| 29686 |                 | space characters or end-of-line.                                                                                                                                                                                                                                                                                                                             |
| 29687 |                 |                                                                                                                                                                                                                                                                                                                                                              |
| 29688 | <i>window</i>   | The current value of the editor option <b>window</b> .                                                                                                                                                                                                                                                                                                       |
| 29689 | <i>word</i>     | In the POSIX locale, <i>vi</i> recognises two kinds of words:                                                                                                                                                                                                                                                                                                |
| 29690 |                 | • A maximal sequence of letters, digits and underscores, delimited at both ends by:                                                                                                                                                                                                                                                                          |
| 29691 |                 | characters other than letters, digits or underscores; the beginning or end of a line; or                                                                                                                                                                                                                                                                     |
| 29692 |                 | the end of the editing buffer.                                                                                                                                                                                                                                                                                                                               |
| 29693 |                 | • A maximal sequence of characters other than letters, digits, underscores or white                                                                                                                                                                                                                                                                          |
| 29694 |                 | space, delimited at both ends by: a letter, digit, underscore or white space; the                                                                                                                                                                                                                                                                            |
| 29695 |                 | beginning or end of a line; or the end of the editing buffer.                                                                                                                                                                                                                                                                                                |
| 29696 | <b>!</b>        | A character that can be appended to the command to modify its operation as detailed                                                                                                                                                                                                                                                                          |
| 29697 |                 | in the individual command descriptions.                                                                                                                                                                                                                                                                                                                      |
| 29698 |                 | After processing a command, if the current line and column indicators have been changed to                                                                                                                                                                                                                                                                   |
| 29699 |                 | specify a character not currently on the screen, the editor will scroll or page the text to bring the                                                                                                                                                                                                                                                        |
| 29700 |                 | needed character onto the screen. If the new current line is within <i>window</i> lines of the previous                                                                                                                                                                                                                                                      |
| 29701 |                 | current line, and the terminal is capable of scrolling in the indicated direction, the text will be                                                                                                                                                                                                                                                          |
| 29702 |                 | scrolled the minimum number of lines necessary to present the needed character. Otherwise, the                                                                                                                                                                                                                                                               |
| 29703 |                 | screen will be redisplayed with the new current line positioned in the middle of the screen.                                                                                                                                                                                                                                                                 |
| 29704 |                 | The following rules specify how exceptions will be handled. When a movement command                                                                                                                                                                                                                                                                          |
| 29705 |                 | would cause a window to display beyond the beginning of the editing buffer, the window                                                                                                                                                                                                                                                                       |
| 29706 |                 | displayed is the window beginning with the first line of the editing buffer. When a movement                                                                                                                                                                                                                                                                 |
| 29707 |                 | command would cause a window to be displayed beyond the end of the editing buffer, the                                                                                                                                                                                                                                                                       |
| 29708 |                 | terminal will be alerted and the command aborted. When the current line is empty, the cursor                                                                                                                                                                                                                                                                 |
| 29709 |                 | will be placed in the first position of the line on the screen. When the current column indicator                                                                                                                                                                                                                                                            |
| 29710 |                 | value is beyond the end of the current line, the cursor will be placed on the last character of the                                                                                                                                                                                                                                                          |
| 29711 |                 | current line, but the current column indicator will be unchanged.                                                                                                                                                                                                                                                                                            |
| 29712 |                 | The following rules specify how position exceptions will be handled. When the current column                                                                                                                                                                                                                                                                 |
| 29713 |                 | indicator is at the beginning (end) of a line and would be set to a position before the beginning                                                                                                                                                                                                                                                            |
| 29714 |                 | (after the end) of the current line, the command will not be executed, the terminal will be alerted,                                                                                                                                                                                                                                                         |
| 29715 |                 | and the current position will remain unchanged. Otherwise, when the current column indicator                                                                                                                                                                                                                                                                 |
| 29716 |                 | would be set to a position before the beginning (after the end) of the current line, it will be set                                                                                                                                                                                                                                                          |
| 29717 |                 | instead to the position of the first (last) character of that line. When the current line indicator                                                                                                                                                                                                                                                          |
| 29718 |                 | would be set to a position before the beginning (after the end) of the editing buffer, the                                                                                                                                                                                                                                                                   |
| 29719 |                 | command will not be executed, the terminal will be alerted, and the current position will remain                                                                                                                                                                                                                                                             |
| 29720 |                 | unchanged.                                                                                                                                                                                                                                                                                                                                                   |

**Page Backwards**

*Synopsis:*     [*count*] <control>-B

Page backward in the file, allowing two lines of overlap, by displaying the window starting at line [*line* - (*count* \* (*window* - 2))].

*Current line:* The last line displayed.

*Current column:* Move to the first non-blank character of the current line or the first character if the line is a blank line.

**Scroll Forward**

*Synopsis:*     [*count*] <control>-D

Scroll forward *count* lines in the file. If *count* is not specified, the same number of lines will be scrolled as by the previous <control>-D or <control>-U command. On the first <control>-D or <control>-U command, the amount scrolled will be one-half the number of lines in a screenful. See the *ex scroll* option.

*Current line:* (*line* + amount scrolled).

*Current column:* Move to the first non-blank character of the current line or the first character if the line is a blank line.

**Scroll Forward by Line**

*Synopsis:*     [*count*] <control>-E

Scroll forward *count* lines, leaving the current line and column as is if possible.

*Current line:* Unchanged unless the current line scrolls off the screen; otherwise, move to the first line displayed.

*Current column:* Unchanged unless the current line scrolls off the screen; otherwise, move to the first character if it is a blank line or the last character of the line if the position of the current column is further into the line than the position of the last column of the line.

**Page Forward**

*Synopsis:*     [*count*] <control>-F

Page forward in the file, allowing two lines of overlap, by displaying the window starting at line [*line* + (*count* \* (*window* - 2))].

*Current line:* The first line displayed.

*Current column:* Move to the first non-blank character of the current line or the first character if the line is a blank line.

**Display Information**

*Synopsis:*     <control>-G

Display an informational message listing the current pathname, current line, number of lines and other unspecified information, as does the *ex* command *file*.

*Current line:* Unchanged.

*Current column:* Unchanged.

29758 **Move Cursor Backwards**29759 *Synopsis:* [ *count* ] <control>-H29760 *Synopsis:* [ *count* ] h

29761 Move the cursor back *count* characters on the current line. No characters will be erased from the  
 29762 screen by this command. In input mode, <control>-H will have the exact same function.

29763 *Current line:* Unchanged.

29764 *Current column:* Set to (*column* – the number of columns occupied by *count* characters ending  
 29765 with the previous current column) or the beginning of the line if *count* is greater than the number  
 29766 of characters preceding the current character in the current line.

29767 **Move Down in Column**29768 *Synopsis:* [ *count* ] <control>-J29769 *Synopsis:* [ *count* ] j29770 *Synopsis:* [ *count* ] <control>-N29771 Move the cursor down *count* lines without changing the current column.

29772 *Current line:* *current line* + *count*. Unchanged if *count* is greater than the number of lines in the  
 29773 buffer following the current line.

29774 *Current column:* Unchanged.29775 **Clear and Redisplay**29776 *Synopsis:* <control>-L

29777 Clear and redisplay the screen.

29778 *Current line:* Unchanged.29779 *Current column:* Unchanged.29780 **Move Cursor Down to Non-blank**29781 *Synopsis:* [ *count* ] <control>-M29782 *Synopsis:* [ *count* ] +29783 Move the cursor down *count* lines to the first non-blank character of that line.29784 *Current line:* *line* + *count*.

29785 *Current column:* Move to the first non-blank character of the current line or the first character if  
 29786 the line is a blank line.

29787 **Move Up in Column**29788 *Synopsis:* [ *count* ] <control>-P29789 *Synopsis:* [ *count* ] k29790 Move the cursor up *count* lines without changing the current column.29791 *Current line:* *line* – *count*.29792 *Current column:* Unchanged.

## 29793 **Redraw Screen**

29794 *Synopsis:*     <control>-R

29795 Redraw the current screen. If any lines have been deleted from the logical screen and flagged as  
 29796 deleted on the terminal using the "@" convention (see the beginning of the EXTENDED  
 29797 DESCRIPTION section), they will be deleted, and the logical screen will be redisplayed. Lines  
 29798 flagged with @ because they do not fit on the terminal display will not be affected.

29799 *Current line:* Unchanged.

29800 *Current column:* Unchanged.

## 29801 **Move Forward with Tabs**

29802 *Synopsis:*     <control>-T

29803 Move the cursor forward *shiftwidth* (see the *ex shiftwidth* option) positions in insert mode by  
 29804 inserting tab characters, followed by space characters, as necessary, if the current cursor position  
 29805 is at the beginning of a line or preceded only by blank characters.

29806 *Current line:* Unchanged.

29807 *Current column:* *column + shiftwidth*.

## 29808 **Scroll Backward**

29809 *Synopsis:*     [*count*] <control>-U

29810 Scroll backward *count* lines in the file. If *count* is not specified, the same number of lines will be  
 29811 scrolled as by the previous <control>-D or <control>-U command. On the first <control>-D or  
 29812 <control>-U command, the amount scrolled will be the value of the *ex scroll* editor option.

29813 *Current line:* (*line* – amount scrolled).

29814 *Current column:* Move to the first non-blank character of the current line or the first character if  
 29815 the line is a blank line.

## 29816 **Escape Next Character**

29817 *Synopsis:*     <control>-V *character*

29818 *Synopsis:*     <control>-Q *character*

29819 Allow the entry of a subsequent *character*, removing any special meaning to the editor it has in  
 29820 input mode (for example, the escape character). The character ^ will be displayed in the current  
 29821 cursor position until the subsequent character is typed, which then replaces the character ^.

29822 *Current line:* Unchanged.

29823 *Current column:* Unchanged.

## 29824 **Delete Word**

29825 *Synopsis:*     <control>-W

29826 Delete the word preceding the cursor (including any blank characters between the end of the  
 29827 word and the current cursor position) in input mode by moving the cursor back to the beginning  
 29828 of the word. No characters will be erased from the screen by this command. Only text entered  
 29829 during the current input mode session and on the current line can be deleted with this  
 29830 command.



29831 *Current line:* Unchanged.

29832 *Current column:* Moved back as described above.

### 29833 **Scroll Backward by Line**

29834 *Synopsis:* [ *count* ] <control>-Y

29835 Scroll backward *count* lines, leaving the current line and column as is, if possible.

29836 *Current line:* Unchanged unless the current line scrolls off the screen; otherwise, the last line  
29837 displayed.

29838 *Current column:* Unchanged unless the current line scrolls off the screen; otherwise, move to the  
29839 first character if it is a blank line or the last character of the line if the position of the current  
29840 column is further into the line than the position of the last column of the line.

### 29841 **Terminate Command or Input Mode**

29842 *Synopsis:* <ESC>

29843 The effects of the escape character depend on the mode and the command being entered, as  
29844 follows:

29845 • In command mode:

29846 — If a line-oriented command ("/", "?", ":" or "!") is being entered, terminate the line-oriented  
29847 command and execute it.

29848 — If only part of a command has been entered, cancel the partial command. A command is  
29849 not considered to be partially entered until at least one non-*count* character has been  
29850 entered. For example, 33c<ESC> is partially entered and silently canceled, whereas  
29851 33<ESC> alerts the terminal (see the next item).

29852 — Otherwise, alert the terminal.

29853 • In input mode: Terminate input mode and return to command mode. At this point any text  
29854 that was deleted in input mode, but that has not yet been erased from the screen, will be  
29855 erased from the screen; the current line will be redisplayed to match exactly the changes  
29856 made in input mode.

29857 *Current line:* When a line-oriented command is being terminated, the current line indicator will  
29858 be set as given for that command. Otherwise, unchanged.

29859 *Current column:* When a line-oriented command is being terminated, the current column  
29860 indicator will be set as given for that command. When terminating input mode, if the cursor is  
29861 not at the beginning of a line, the current column position will be set to (*column* – the width of  
29862 the last character inserted). Otherwise, unchanged.

### 29863 **Move Cursor Forward**

29864 *Synopsis:* [ *count* ] <space>

29865 *Synopsis:* [ *count* ] 1 (ell)

29866 Move the cursor forward *count* characters without changing the current line.

29867 *Current line:* Unchanged.

29868 *Current column:* (*current column* + the width of the next *count* characters) or the end of the line if  
29869 *count* is greater than the number of characters in the line following the current column.

## Replace Text with Results from Shell Command

**Synopsis:** [count] ! motion shell-commands <newline>

Pass the lines specified by *count* and *motion* as standard input to a shell command, for the program named in the *SHELL* environment variable, and replace those lines with the standard output of the shell command. An implementation may consider a *motion* command that does not result in an integral number of lines to be an error. After the *motion* is entered, the text of the command will be prompted for on the last line of the display as described at the beginning of the EXTENDED DESCRIPTION section. A warning will be issued if the buffer has been changed since the last write.

Within the text of *command*, "%" and "#" will be expanded as pathnames (the current and alternative pathnames, respectively), and "!" will be replaced with the text of the previous "!" or :! command. (Thus, !! will repeat the previous "!" command.) If no "!" or :! command has yet been executed, an informational message will be displayed stating the problem.

The special meanings of "%", "#" and "!" can be overridden by escaping them with a backslash character. This command is affected by the *ex* editor options **autowrite** and **writeln**.

**Current line:** The first line in *range*.

**Current column:** The first column of the replaced text.

## Move Cursor to End-of-line

**Synopsis:** [count] \$

Move the cursor forward to the end of a line. The *count* argument specifies the number of lines, including the current line, to move forward.

**Current line:** *line* + *count* - 1.

**Current column:** *end-of-line*.

## Move to Matching Character

**Synopsis:** %

Move the cursor to the parenthesis or curly brace matching the parenthesis or curly brace at the current position or on the current line forward from the current position. Matching will be determined as follows: for a left parenthesis (respectively curly brace) the editing buffer will be searched forward until either a left or right parenthesis is found. If a left parenthesis is found, a counter is incremented by one and if a right parenthesis is found, a counter is decremented by one, and the search continues (until the end of the editing buffer is found). If a right parenthesis is found when the count is less than or equal to zero, it is the matching parenthesis. For a right parenthesis (respectively curly brace) the editing buffer will be searched backward (to the beginning of the editing buffer), and each right parenthesis increments the counter, and each left parenthesis decrements it. When searching for parentheses, curly braces are not counted, and vice versa. If there is no matching character, the terminal will be alerted and the current position will remain unchanged.

**Current line:** Move to the line containing the matching character, as described above.

**Current column:** Move to the column containing the matching character, as described above.

## 29909 **Repeat Substitution**

29910 *Synopsis:*     &

29911 Repeat the previous substitution command (equivalent to the *ex* & command) using the current  
29912 line as the target. Flags specified on the previous substitution command will be ignored by this  
29913 command.

29914 *Current line:* Unchanged.

29915 *Current column:* Unchanged if the previous current column indicator value was *end-of-line*;  
29916 otherwise, move to the first non-blank character of the current line or the first character if the  
29917 line is a blank line.

## 29918 **Return to Previous Context at Beginning of Line**

29919 *Synopsis:*     '

29920 *Synopsis:*     ' *letter*

29921 Return to the previous context when followed by a ' character, if the context still exists, placing  
29922 the cursor at the first non-blank character of the line or the last character if the line is a blank line.  
29923 If the previous context no longer exists, the ' command will have no effect. When followed by a  
29924 lower-case letter from the POSIX locale, return to the line marked with that letter (see the *m*  
29925 command), at the first non-blank character in the line.

29926 When used with an operator such as *d* to specify an extent of text, the operation takes place over  
29927 complete lines. (See also **Return to Previous Context**.)

29928 *Current line:* Move to the line from the previous context.

29929 *Current column:* Move to the first non-blank character of the current line or the last character if  
29930 the line is a blank line.

## 29931 **Return to Previous Context**

29932 *Synopsis:*     `

29933 *Synopsis:*     ` *letter*

29934 When followed by a ` character, return to the previous context, placing the cursor at the  
29935 character position marked. (The previous context will be set whenever a non-relative move is  
29936 made.) When followed by a lower-case letter from the POSIX locale, return to the line marked  
29937 with that letter (see the *m* command), at the character position marked.

29938 When used with an operator such as *d* to specify an extent of text, the operation takes place from  
29939 the exact marked place to the current position within the line. (See also **Return to Previous**  
29940 **Context at Beginning of Line**.)

29941 *Current line:* Move to the line from the previous context.

29942 *Current column:* Set to the position of the character marked from the previous context.

## 29943 **Return to Previous Section**

29944 *Synopsis:*     [ [

29945 Back up to the previous section boundary. This command is affected by the *ex sections* option.  
 29946 Note that the brackets in this command are part of the command syntax itself and are not  
 29947 metacharacters.

29948 EX If the *lisp* option is set, a section boundary is also identified by a line with a leading "(".

29949 *Current line:* Move to the previous line that is a section boundary or to the first line of the editing  
 29950 buffer if no more section boundaries exist preceding the current line.

29951 *Current column:* Move to the first non-blank character of the current line or the last character if  
 29952 the line is a blank line.

## 29953 **Move to Next Section**

29954 *Synopsis:*     ] ]

29955 Move forward to a section boundary. This command is affected by the *ex sections* option. Note  
 29956 that the brackets in this command are part of the command syntax itself and are not  
 29957 metacharacters.

29958 EX If the *lisp* option is set, a section boundary is also identified by a line with a leading "(".

29959 *Current line:* Move to the next line that is a section boundary or to the last line of the editing  
 29960 buffer if no more section boundaries exist following the current line.

29961 *Current column:* Move to the first non-blank character of the current line or the last character if  
 29962 the line is a blank line.

## 29963 **Move to First Non-blank Position on Current Line**

29964 *Synopsis:*     ^

29965 Move to the first non-blank position on the current line.

29966 *Current line:* Unchanged.

29967 *Current column:* Move to the first non-blank character of the current line or the last character if  
 29968 the line is a blank line.

## 29969 **Move Back to Beginning of Sentence**

29970 *Synopsis:*     [ *count* ] (

29971 Move backward to the beginning of a sentence. A *count* will move back that many sentences.

29972 EX If the *lisp* option is set, a *lisp* s-expression is considered a sentence for this command.

29973 *Current line:* Move to the line containing the beginning of the sentence.

29974 *Current column:* Move to the first non-blank character of the sentence.

### 29975 **Move Forward to Beginning of Sentence**

29976 *Synopsis:* [ *count* ] )

29977 Move forward to the beginning of a sentence. A *count* will move forward that many sentences.

29978 EX If the *lisp* option is set, a *lisp* s-expression is considered a sentence for this command.

29979 *Current line:* Move to the line containing the beginning of the sentence.

29980 *Current column:* Move to the first non-blank character of the sentence.

### 29981 **Move Back to Preceding Paragraph**

29982 *Synopsis:* [ *count* ] {

29983 Move back to the beginning of the preceding paragraph. A *count* specifies the number of  
29984 paragraphs to move backward. This command is affected by the *ex* **paragraph** option.

29985 *Current line:* Move to the line containing the beginning of the previous paragraph.

29986 *Current column:* Move to the first non-blank character of the current line.

### 29987 **Move Forward to Next Paragraph**

29988 *Synopsis:* [ *count* ] }

29989 Move forward to the beginning of the next paragraph. A *count* specifies the number of  
29990 paragraphs to move forward. This command is affected by the *ex* **paragraph** option.

29991 *Current line:* Move to the line containing the beginning of the next paragraph.

29992 *Current column:* Move to the first non-blank character of the current line.

### 29993 **Move to Specific Column Position**

29994 *Synopsis:* [ *count* ] |

29995 Place the cursor in the column position identified by *count* (if that position exists on the line). If  
29996 *count* is omitted, it defaults to 1. If the length of the current line is less than *count*, the cursor will  
29997 be placed at the end of the current line. If the column position is spanned by a tab or a wide  
29998 character, the cursor will be placed on the character following the *count*-th column position.

29999 *Current line:* Unchanged.

30000 *Current column:* Move to the column position represented by *count*.

### 30001 **Reverse Find Character**

30002 *Synopsis:* [ *count* ] ,

30003 Reverse the last f, F, t or T command, looking the other way in the current line. A *count* will be  
30004 equivalent to repeating the search that many times.

30005 *Current line:* Unchanged.

30006 *Current column:* Move to the first column position of the next character found.

### 30007 **Move to First Non-blank of Previous Line**

30008 *Synopsis:* [ *count* ] -

30009 Move to the first non-blank character in the previous line. A *count* specifies how many lines to  
30010 move back.

30011 *Current line:* *line* - *count*

30012 *Current column:* Move to the first non-blank character of the current line or the last character if  
30013 the line is a blank line.

### 30014 **Repeat**

30015 *Synopsis:* [ *count* ] .

30016 Repeat the last command that changed the buffer. A *count* will be passed on to the command  
30017 being repeated.

30018 *Current line:* Dependent on the results of the execution of the last command that changed the  
30019 buffer.

30020 *Current column:* Dependent on the results of the execution of the last command that changed the  
30021 buffer.

### 30022 **Find Regular Expression**

30023 *Synopsis:* /

30024 Prompt the user to enter a string on the last line on the screen, interpret it as a regular expression  
30025 (see **Regular Expressions in ex** on page 324), and scan forward for the next occurrence of a  
30026 matching string. The search will begin when a newline character or an escape character is  
30027 entered to terminate the pattern; it can be prematurely terminated with an interrupt.

30028 When used with an operator to specify an extent of text, the defined region is from the current  
30029 cursor position to the beginning of the matched string. Whole lines can be specified by giving an  
30030 offset from the matched line (using a closing / followed by a +*n* or -*n*).

30031 *Current line:* Move to the line in which the first (or next) regular expression match occurred.

30032 *Current column:* Move to the column of the first character of the matched string.

### 30033 **Move to First Character in Line**

30034 *Synopsis:* 0 (zero)

30035 Move to the first character on the current line. (The zero will not be interpreted as a command  
30036 when it is preceded by a non-zero digit.)

30037 *Current line:* Unchanged.

30038 *Current column:* Column position 1.

30039 **Execute an ex Command**30040 *Synopsis:*       :

30041 Execute an **ex** command. The implementation need not support an **ex** command that enters  
 30042 input mode (that is, **append**, **change** or **insert**). The **:** character, as well as the entered command,  
 30043 will be displayed on the bottom line. The command will be executed when the input is  
 30044 terminated by entering a newline character or an escape character. Typing a <control>-V (or  
 30045 <control>-Q) character will escape the following character, allowing its literal value to be  
 30046 entered.

30047 **Note:** This special property of <control>-V and <control>-Q is only effective in visual or open  
 30048 modes and in **ex** commands called from visual or open mode with the **:** command.

30049 If the **ex** command causes the screen to be scrolled or causes an escape to the shell, **vi** will  
 30050 display a message indicating that it is waiting for a newline character. When a newline or an  
 30051 escape character is entered, the screen will be refreshed.

30052 When executing an individual **ex** command by entering **:**, it is not possible to enter a newline  
 30053 character as part of the command because it is considered the end of the command. As  
 30054 explained in **Replacement Strings in ex** on page 324, a <control>-M preceded by a <control>-V  
 30055 (or <control>-Q) escape character in a regular expression substitution is mapped into a newline  
 30056 character. A different approach is to enter **ex** command mode by using the **vi Q** command (and  
 30057 later resuming visual mode with the **ex vi** command). In **ex** command mode, the single-line  
 30058 limitation does not exist. So, for example, the following is valid:

```
30059 Q
30060 s/break here/break\
30061 here/
30062 vi
```

30063 *Current line:* Dependent on the results of the execution of the **ex** command.

30064 *Current column:* Dependent on the results of the execution of the **ex** command.

30065 **Repeat Find**30066 *Synopsis:*       [*count*] ;

30067 Repeat the last character find using **f**, **F**, **t** or **T**. A *count* will move the cursor to (or next to) the  
 30068 *count*-th occurrence of the character.

30069 *Current line:* Unchanged.

30070 *Current column:* Move to the column containing the character being searched for. If no matching  
 30071 character is found, it will be unchanged.

30072 **Shift Left**30073 *Synopsis:*       [*count*] < *motion*

30074 Shift lines left one *shiftwidth* (see the **ex shiftwidth** option). The command must be followed by a  
 30075 motion command to specify lines. A *count* will be passed through to the motion command.

30076 When *motion* is "<", it will shift the current line (or *count* lines starting at the current one).

30077 *Current line:* Unchanged.

30078 *Current column:* Move to the first non-blank character of the line or the last character if the line is  
 30079 a blank line.

### 30080 **Shift Right**

30081 *Synopsis:*     [ *count* ] > *motion*

30082 Shift lines right one *shiftwidth* (see the *ex shiftwidth* option). The command must be followed by  
30083 a motion command to specify lines. A *count* will be passed through to the motion command.

30084 When *motion* is ">", it will shift the current line (or *count* lines starting at the current one). |

30085 Empty lines will not be changed.

30086 *Current line:* Unchanged.

30087 *Current column:* Move to the first non-blank character of the line or the last character if the line is  
30088 a blank line.

### 30089 **Scan Backwards for Regular Expression**

30090 *Synopsis:*     ?

30091 Scan backwards, the reverse of /.

30092 *Current line:* Move to the line in which the next succeeding regular expression match occurred.

30093 *Current column:* Move to the column of the first character of the matched string.

### 30094 **Execute**

30095 *Synopsis:*     @*buffer*

30096 Execute each line of the named buffer as one or more *vi* commands (including *ex* commands, as  
30097 described in **Execute an ex Command** on page 815). If the buffer name is @, then the last buffer  
30098 executed is used; if there is no last buffer, an error will occur. The text of a macro may contain  
30099 an @ character calling another macro; however, recursively calling the same macro produces  
30100 undefined behaviour. A *vi* error will terminate all currently executing macros. All changes  
30101 made during a macro call will be treated as a unit and can be undone with a single u command.

30102 *Current line:* Dependent on the results of the execution of the *vi* commands.

30103 *Current column:* Dependent on the results of the execution of the *vi* commands.

### 30104 **Reverse Case**

30105 *Synopsis:*     [ *count* ] ~

30106 Reverse the case of the *count* characters beginning at the current cursor position. Lower-case  
30107 alphabetic characters will be changed to upper-case and upper-case characters changed to  
30108 lower-case. This command will have no effect on non-alphabetic characters.

30109 *Current line:* Unchanged.

30110 *Current column:* Move to the next column unless it is on the last character of the line, in which  
30111 case it will remain unchanged.



|          |                                                                                                                    |
|----------|--------------------------------------------------------------------------------------------------------------------|
| 30112    | <b>Reindent</b>                                                                                                    |
| 30113 EX | <i>Synopsis:</i> [ <i>count</i> ]=[ <i>motion</i> ]                                                                |
| 30114    | If the <i>lisp</i> option is set, reindents the specified lines, as though they were typed in with <b>lisp</b> and |
| 30115    | <b>autoindent</b> set.                                                                                             |
| 30116    | <i>Current line:</i> Unchanged.                                                                                    |
| 30117    | <i>Current column:</i> Move to the first non-blank character of the line or the last character if the line is      |
| 30118    | a blank line.                                                                                                      |
| 30119    | <b>Append</b>                                                                                                      |
| 30120    | <i>Synopsis:</i> [ <i>count</i> ] a                                                                                |
| 30121    | Enter input mode, appending the entered text after the current cursor position. A <i>count</i> will                |
| 30122    | cause the inserted text up to the first newline character to be replicated that many times; if a                   |
| 30123    | newline character appears in the inserted text, only one occurrence of it and any characters                       |
| 30124    | following it will be inserted. For example, if the current line is <b>abc</b> , the command:                       |
| 30125    | 03afoo<newline>bar                                                                                                 |
| 30126    | changes that line to:                                                                                              |
| 30127    | afoofoofoo                                                                                                         |
| 30128    | barbc                                                                                                              |
| 30129    | <i>Current line:</i> Unchanged.                                                                                    |
| 30130    | <i>Current column:</i> <i>column</i> + 1.                                                                          |
| 30131    | <b>Append at End-of-line</b>                                                                                       |
| 30132    | <i>Synopsis:</i> [ <i>count</i> ] A                                                                                |
| 30133    | Append <i>count</i> copies of the input text at the end of the current line, equivalent to \$[ <i>count</i> ] a.   |
| 30134    | <i>Current line:</i> Unchanged.                                                                                    |
| 30135    | <i>Current column:</i> See the a command.                                                                          |
| 30136    | <b>Move Backward to Preceding Word</b>                                                                             |
| 30137    | <i>Synopsis:</i> [ <i>count</i> ] b                                                                                |
| 30138    | Move the cursor backward to the beginning of a word by repeating the following algorithm                           |
| 30139    | <i>count</i> times: If the current position is at the beginning of a word, the current position will move          |
| 30140    | to the first character of the preceding word. Otherwise, the current position will move to the                     |
| 30141    | first character of the word at the current position. If no preceding word exists on the current                    |
| 30142    | line, the current position will move to the first character of the last word on the first preceding                |
| 30143    | line that contains a word. For this command, an empty or blank line will be considered to                          |
| 30144    | contain exactly one word.                                                                                          |
| 30145    | <i>Current line:</i> Set to the line containing the word selected.                                                 |
| 30146    | <i>Current column:</i> Set to the first character of the word selected.                                            |

### 30147 **Move Backward to Preceding Bigword**

30148 *Synopsis:*     [*count*] B

30149 Move the cursor backward to the beginning of a bigword by repeating the following algorithm  
30150 *count* times: If the current position is at the beginning of a bigword or the character at the  
30151 current position cannot be part of a bigword, the current position will move to the first character  
30152 of the preceding bigword. Otherwise, the current position will move to the first character of the  
30153 bigword at the current position. If no preceding bigword exists on the current line, the current  
30154 position will move to the first character of the last bigword on the first preceding line that  
30155 contains a bigword. For this command, an empty or blank line is considered to contain exactly  
30156 one bigword.

30157 *Current line:* Set to the line containing the bigword selected.

30158 *Current column:* Set to the first character of the bigword selected.

### 30159 **Change**

30160 *Synopsis:*     [*buffer*][*count*] c *motion*

30161 Delete the specified region of text and enter input mode to replace it with the entered text. If  
30162 more than part of a single line is affected, the deleted text will be saved in the numeric buffers. If  
30163 only part of the current line is affected, then the last character to be deleted will be marked with  
30164 a \$. A *count* will be passed through to the motion command.

30165 *Current line:* The current line and column position are dependent on the *motion* command  
30166 following the c. For example, **cw** changes a word, **c[[** changes from the beginning of the current  
30167 section, and so on.

30168 *Current column:* See *Current line*, above

### 30169 **Change to End-of-line**

30170 *Synopsis:*     [*buffer*][*count*] C

30171 Change text from the current position to the end-of-line (*line* + *count* – 1); equivalent to:

30172     [*buffer*][*count*] c\$

30173 *Current line:* See the c command.

30174 *Current column:* See the c command.

### 30175 **Delete**

30176 *Synopsis:*     [*buffer*][*count*] d *motion*

30177 Delete the specified region of text. If more than part of a line is affected, the text will be saved in  
30178 the numeric buffers as well as any named *buffer*. A *count* will be passed through to the motion  
30179 command.

30180 *Current line:* The current line and column position are dependent on the *motion* command  
30181 following the d. For example, **dw** deletes a word, **d\$** deletes to the end of the line, and so on.

30182 *Current column:* See *Current line*, above.

### 30183 **Delete to End-of-line**

30184 *Synopsis:*     [*buffer*] D

30185 Delete the text from the current position to the end of the current line; equivalent to:

30186         [*buffer*] d\$

30187 *Current line:* Unchanged.

30188 *Current column:* Set to maximum of *column* – 1 or 1.

### 30189 **Move to End-of-word**

30190 *Synopsis:*     [*count*] e

30191 Move the cursor forward to the end of a word, by repeating the following algorithm *count* times:  
 30192 If the current position is the end of a word, the current position will move to the last character of  
 30193 the following word. Otherwise, the current position will move to the last character of the word  
 30194 at the current position. If no succeeding word exists on the current line, the current position will  
 30195 move to the last character of the first word on the next following line that contains a word. For  
 30196 this command, an empty or blank line is considered to contain exactly one word.

30197 *Current line:* Set to the line containing the word selected.

30198 *Current column:* Set to the last character of the word selected.

### 30199 **Move to End-of-bigword**

30200 *Synopsis:*     [*count*] E

30201 Move the cursor forward to the end of a bigword, by repeating the following algorithm *count*  
 30202 times: If the current position is the end of a bigword or the character at that position cannot be  
 30203 part of a bigword, the current position will move to the last character of the following bigword.  
 30204 Otherwise, the current position will move to the last character of the bigword at the current  
 30205 position. If no succeeding bigword exists on the current line, the current position will move to  
 30206 the last character of the first bigword on the next following line that contains a bigword.

30207 For this command, an empty or blank line is considered to contain exactly one bigword.

30208 *Current line:* Set to the line containing the bigword selected.

30209 *Current column:* Set to the last character of the bigword selected.

### 30210 **Find Character in Current Line (Forward)**

30211 *Synopsis:*     [*count*] f *character*

30212 Scan the rest of the current line for the single character *character* and move the cursor to it if it is  
 30213 found. A *count* will repeat the find that many times.

30214 *Current line:* Unchanged.

30215 *Current column:* Set to the column position containing the character that was scanned for.  
 30216 Unchanged if insufficient characters were found to satisfy the *count*.

### 30217 **Find Character in Current Line (Reverse)**

30218 *Synopsis:*     [ *count* ] F *character*

30219 Scan backwards in the current line for the single character *character*, moving the cursor to it if it  
30220 is found. A *count* will be equivalent to repeating the search that many times. If the search is  
30221 unsuccessful, the terminal will be alerted and the current column position will remain  
30222 unchanged.

30223 *Current line:* Unchanged.

30224 *Current column:* Set to the last character found moving backwards.

### 30225 **Move to Line**

30226 *Synopsis:*     [ *count* ] G

30227 Move the cursor to line *count*, or to the last line of the editing buffer if *count* is not specified.

30228 *Current line:* *count*, if specified; otherwise, the last line.

30229 *Current column:* Move to the first non-blank character of the current line or the last character if  
30230 the line is a blank line.

### 30231 **Move to Top of Screen**

30232 *Synopsis:*     [ *count* ] H

30233 Move the cursor to the line (*count* – 1) lines from the top of the current window (that is, the  
30234 *count*-th line, one-based, currently displayed).

30235 *Current line:* *count* on the screen, as described above.

30236 *Current column:* Move to the first non-blank character of the current line or the last character if  
30237 the line is a blank line.

### 30238 **Insert Before Cursor**

30239 *Synopsis:*     [ *count* ] i

30240 Enter input mode, inserting the entered text before the cursor. (See also the a command.) A  
30241 *count* will cause the inserted text up to the first newline character to be replicated that many  
30242 times; if a newline character appears in the inserted text, only one occurrence of it and any  
30243 characters following it will be inserted.

30244 *Current line:* Unchanged.

30245 *Current column:* Unchanged.

### 30246 **Insert at Beginning of Line**

30247 *Synopsis:*     [ *count* ] I

30248 Enter input mode at the beginning of a line. This command behaves identically to the ^[ *count* ] i  
30249 command.

30250 *Current line:* Unchanged.

30251 *Current column:* Move to the first non-blank character of the current line or the last character if  
30252 the line is a blank line.

30253 **Join**30254 *Synopsis:*     [*count*] J

30255 Join the current line with the next one, supplying appropriate spacing (in the POSIX locale): one  
 30256 space character between words, two space characters after a period, and no space characters at  
 30257 all when the last character of the first line is a blank character or when the first character of the  
 30258 next line is "). In the case of lines ending with blank characters, the blanks characters will be  
 30259 retained, no spaces will be added, and the lines will be joined together. A *count* will cause that  
 30260 many lines (minimum 2) to be joined, rather than 2. A count of 1 will be treated as 2.

30261 *Current line:* Unchanged.30262 *Current column:* Move to the character after the last character of the original line.30263 **Move to Bottom of Screen**30264 *Synopsis:*     [*count*] L

30265 Move the cursor to the first non-blank character of the last line on the screen. A *count* will move  
 30266 to that line counting from the bottom. When used with an operator, whole lines are affected.

30267 *Current line:* *last line – count*

30268 *Current column:* Move to the first non-blank character of the current line or the last character if  
 30269 the line is a blank line.

30270 **Mark Position**30271 *Synopsis:*     m *letter*

30272 Mark the current position of the cursor with a single POSIX locale lower-case letter *letter*. The  
 30273 exact position is referred to by ``letter`; the line is referred to by `'letter`.

30274 *Current line:* Unchanged.30275 *Current column:* Unchanged.30276 **Move to Middle of Screen**30277 *Synopsis:*     M

30278 Move the cursor to the middle line on the screen, at the first non-blank position on the line.

30279 *Current line:* Move to a line approximately in the middle of the display window.

30280 *Current column:* Move to the first non-blank character of the current line or the last character if  
 30281 the line is a blank line.

30282 **Repeat Regular Expression Find (Forward)**30283 *Synopsis:*     n

30284 Repeat the last / or ? scanning command.

30285 *Current line:* Set to the line containing the character string found by the scan. Unchanged if no  
 30286 match was found.

30287 *Current column:* Move to the first character of the character string found by the scan.  
 30288 Unchanged if no match was found.

### 30289 **Repeat Regular Expression Find (Reverse)**

30290 *Synopsis:*     N

30291 Scan for the next match of the last pattern given to / or ?, but in the reverse direction; this is the  
30292 reverse of **n**.

30293 *Current line:* Move to the matched line if a match is found or unchanged if no match is found.

30294 *Current column:* Move to the first column position of the matched string or unchanged if no  
30295 match is found.

### 30296 **Insert Empty Line Below**

30297 *Synopsis:*     o

30298 Open a line below the current line and enter input mode.

30299 *Current line:* The newly-created line, (*line*+1).

30300 *Current column:* If the editor option **autoindent** is not set, move to the first column; otherwise, as  
30301 in the description of the *ex* **autoindent** option.

### 30302 **Insert Empty Line Above**

30303 *Synopsis:*     O

30304 Open a new line above the current line and enter input mode.

30305 *Current line:* Unchanged.

30306 *Current column:* If the editor option **autoindent** is not set, move to the first column; otherwise, as  
30307 described in the description of **autoindent** in *ex*.

### 30308 **Put from Buffer Following**

30309 *Synopsis:*     [*buffer*] p

30310 Insert text from buffer *buffer* following the current column or current line, as described for the P  
30311 command. If *buffer* is omitted, the unnamed buffer is used.

30312 *Current line:* (*Current line*+1) if the buffer contains whole lines; otherwise, unchanged.

30313 *Current column:* First non-blank character of the inserted text if the buffer contained whole lines;  
30314 otherwise, the last character of the inserted text.

### 30315 **Put from Buffer Before**

30316 *Synopsis:*     [*buffer*] P

30317 Put the last deleted text back before and above the cursor. The text will go back as whole lines  
30318 above the cursor if it was deleted as whole lines. Otherwise, the text will be inserted just before  
30319 the cursor.

30320 The command can be preceded by a named buffer specification ("x), to retrieve the contents of  
30321 the buffer.

30322 *Current line:* Unchanged.

30323 *Current column:* Move to the last column position of the inserted characters. If the buffer  
30324 contains whole lines, the current column will be the first non-blank character of the inserted  
30325 characters.

30326 **Enter ex Mode**30327 *Synopsis:*     Q30328 Quit from *vi* and enter *ex* command mode.30329 *Current line:* Unchanged.30330 *Current column:* Unchanged.30331 **Replace Character**30332 *Synopsis:*     [*count*] r *character*

30333 Replace a character on the screen with the character entered. If *count* is specified, the single  
 30334 character entered will replace the current character and each of the next *count* – 1 characters in  
 30335 the line. (For example, 7rx changes the next seven characters to be **xxxxxxx**). Unlike the R  
 30336 command, entering a newline character will replace the character with a newline character;  
 30337 however, when used with *count*, the next *count* characters in the line will be replaced by a single  
 30338 newline character.

30339 *Current line:* Unchanged unless the replacement character is a newline character, in which case  
 30340 the current line will be incremented by 1.

30341 *Current column:* Set to the last column position occupied by the replacement character, unless  
 30342 the replacement character is a newline character, in which case the current column position will  
 30343 be 1.

30344 **Replace Characters**30345 *Synopsis:*     R

30346 Replace characters on the screen with characters entered. If the end of the existing line is  
 30347 encountered, it will be as if insert mode was entered at that point. If a newline character is  
 30348 entered before the end of the existing line, it will be entered into the editing buffer as if it was  
 30349 inserted, and replacement will continue on the newly created line. No other line but the one in  
 30350 which the R command was given will be affected.

30351 *Current line:* Set to the line at the end of the replaced text.

30352 *Current column:* Set to the end of the replaced text, if text was replaced. If no text was replaced,  
 30353 set to the column position of the character after which text would have been replaced if that  
 30354 character is on the current line; otherwise, 1.

30355 **Substitute Character**30356 *Synopsis:*     [*buffer*][*count*] s

30357 Substitute *count* characters in the current line starting with the current column position;  
 30358 equivalent to:

30359       [*buffer*][*count*] c<space>30360 *Current line:* Unchanged.30361 *Current column:* Unchanged.

### 30362 **Substitute lines**

30363 *Synopsis:*     [*buffer*][*count*] S

30364 Change whole lines (same as **cc**). A *count* will change that many lines; *count* lines will be deleted  
30365 and *vi* will enter input mode.

30366 *Current line:* Unchanged.

30367 *Current column:* Set to column position 1 or to the position indicated by the previous line if  
30368 **autoindent** is set.

### 30369 **Move Cursor to Before Character (Forward)**

30370 *Synopsis:*     [*count*] t *character*

30371 Move the cursor forward within the current line to the character position just before a  
30372 subsequent occurrence of *character*. A *count* will place the cursor just before the *count*-th  
30373 occurrence of the character when searching the line forwards. If the search is unsuccessful, the  
30374 terminal will be alerted and the cursor position will remain unchanged.

30375 *Current line:* Unchanged.

30376 *Current column:* Move to the position before an occurrence of *character* following *column* (non-  
30377 inclusive).

### 30378 **Move Cursor to After Character (Reverse)**

30379 *Synopsis:*     [*count*] T *character*

30380 Scan backwards in the current line for the single character *character* and if found, place the cursor  
30381 just after that character. A *count* will place the cursor just after the *count*-th occurrence of the  
30382 character when searching the line backwards. If *count*-th occurrences of the character do not  
30383 exist, the terminal will be alerted and the current column position will remain unchanged.

30384 *Current line:* Unchanged.

30385 *Current column:* Set to the column after the scanned character. Unchanged if the character was  
30386 not found.

### 30387 **Undo**

30388 *Synopsis:*     u

30389 Reverse the last change made to the current buffer. If repeated, the command will alternate  
30390 between these two states, thus is its own inverse. When used after an insert that inserted text on  
30391 more than one line, the lines will be saved in the numeric named buffers.

30392 *Current line:* Move to the position of the first line changed, if the reversal affects only one line or  
30393 represents an addition or change; otherwise, move to the line preceding the deleted text.

30394 *Current column:* Move to the first non-blank character on the updated current line.



### 30395 **Undo Current Line**

30396 *Synopsis:*     `U`

30397 Restore the current line to its state before the cursor was last moved to it.

30398 *Current line:* Unchanged.

30399 *Current column:* Set to column position 1 or to the position indicated by the previous line if  
30400 **autoindent** is set.

### 30401 **Move to Beginning of Word**

30402 *Synopsis:*     `[count] w`

30403 Move the cursor forward to the beginning of a word by repeating the following algorithm *count*  
30404 times: If the current position is at the beginning of a word, the current position will move to the  
30405 first character of the next word. If no subsequent word exists on the current line, the current  
30406 position will move to the first character of the first word on the first following line that contains  
30407 a word.

30408 For this command, an empty or blank line is considered to contain exactly one word.

30409 *Current line:* Set to the line containing the word selected.

30410 *Current column:* Set to the first character of the word selected.

### 30411 **Move to Beginning of Bigword**

30412 *Synopsis:*     `[count] W`

30413 Move the cursor forward to the beginning of a bigword by repeating the following algorithm  
30414 *count* times: If the current position is within a bigword or the character at that position cannot  
30415 be part of a bigword, the current position will move to the first character of the next bigword. If  
30416 no subsequent bigword exists on the current line, the current position will move to the first  
30417 character of the first bigword on the first following line that contains a bigword. For this  
30418 command, an empty or blank line is considered to contain exactly one bigword.

30419 *Current line:* Set to the line containing the bigword selected.

30420 *Current column:* Set to the first character of the bigword selected.

### 30421 **Delete Character at Cursor**

30422 *Synopsis:*     `[buffer][count] x`

30423 Delete *count* characters in the current line starting with the current column position; equivalent  
30424 to:

30425     `[buffer][count] dl`

30426 If the number of characters to be deleted is greater than or equal to the number of characters to  
30427 the end of the line, delete all of the characters from the current position to the end of the line and  
30428 move the cursor to the new last character on the line.

30429 *Current line:* Unchanged.

30430 *Current column:* Set to the greater of (*current column* – the width of the *count* deleted characters)  
30431 or 1.

### 30432 **Delete Character Before Cursor**

30433 *Synopsis:*     [*buffer*][*count*] X

30434 Delete the character before the cursor; equivalent to:

30435         [*buffer*][*count*] dh

30436 A *count* will repeat the effect, but only characters on the current line will be deleted.

30437 *Current line:* Unchanged.

30438 *Current column:* Set to the greater of (*current column* – the width of the *count* deleted characters)  
30439 or 1.

### 30440 **Yank**

30441 *Synopsis:*     [*buffer*][*count*] y *motion*

30442 Copy (yank) the area of text specified by *count* and *motion* into the unnamed buffer, as well as  
30443 into any named buffer specified by *buffer*.

30444 *Current line:* Unchanged.

30445 *Current column:* Unchanged.

### 30446 **Yank Current Line**

30447 *Synopsis:*     [*buffer*][*count*] Y

30448 Copy (yank) *count* lines starting with the current line into the unnamed buffer, as well as into  
30449 any named buffer specified by *buffer*; equivalent to a:

30450         [*buffer*][*count*] yy

30451 command.

30452 *Current line:* Unchanged.

30453 *Current column:* Unchanged.

### 30454 **Redraw Window**

30455 *Synopsis:*     [*count1*] z [*count2*] *character*

30456 Redraw the screen with a window *count2* lines long containing line *count1* placed as specified by  
30457 *character*: a newline character specifies the top of the screen, "." specifies the center of the screen,  
30458 and "-" specifies the bottom of the screen. If *count1* is not specified, it defaults to the current line;  
30459 if *count2* is not specified, it defaults to *window*.

30460 *Current line:* *count1*, if specified; otherwise, unchanged.

30461 *Current column:* Move to the first non-blank character of the current line or the first character if  
30462 the line is a blank line.

**30463 Exit**

30464 *Synopsis:*     *ZZ*

30465 Exits the editor, writing out the buffer if it was changed since the last write to any file. This  
30466 command behaves identically to the *ex* command **xit**.

30467 The uses described for <control>-V can also be accomplished with <control>-Q, which is useful  
30468 on terminals that use <control>-V for the down-arrow function. However, most historical  
30469 implementations use <control>-Q for the **termios** START character, so the editor will generally  
30470 not receive the <control>-Q unless *stty -ixon* mode is set. Any of the command characters  
30471 described in this specification can be made ineffective by their selection as **termios** control  
30472 characters, using the *stty* utility or other methods described in the **XBD** specification, **Chapter 9**,  
30473 **General Terminal Interface**.

**30474 EXIT STATUS**

30475 The following exit values are returned:

30476     0 Successful completion.

30477     >0 An error occurred.

**30478 CONSEQUENCES OF ERRORS**

30479 Default.

**30480 APPLICATION USAGE**

30481 None.

**30482 EXAMPLES**

30483 None.

**30484 FUTURE DIRECTIONS**

30485 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
30486 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
30487 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
30488 finalised.

**30489 SEE ALSO**

30490 *ex*.

**30491 CHANGE HISTORY**

30492 First released in Issue 2.

**30493 Issue 4**

30494 Aligned with the ISO/IEC 9945-2: 1993 standard.

**30495 Issue 5**

30496 FUTURE DIRECTIONS section added.

30497 **NAME**

30498           wait — await process completion

30499 **SYNOPSIS**30500           wait [*pid*...]30501 **DESCRIPTION**

30502           When an asynchronous list (see Section 2.9.3 on page 50) is started by the shell, the process ID of  
 30503           the last command in each element of the asynchronous list becomes known in the current shell  
 30504           execution environment; see Section 2.12 on page 63.

30505           If the *wait* utility is invoked with no operands, it will wait until all process IDs known to the  
 30506           invoking shell have terminated and exit with a zero exit status.

30507           If one or more *pid* operands are specified that represent known process IDs, the *wait* utility will  
 30508           wait until all of them have terminated. If one or more *pid* operands are specified that represent  
 30509           unknown process IDs, *wait* will treat them as if they were known process IDs that exited with  
 30510           exit status 127. The exit status returned by the *wait* utility will be the exit status of the process  
 30511           requested by the last *pid* operand.

30512           The known process IDs are applicable only for invocations of *wait* in the current shell execution  
 30513           environment.

30514 **OPTIONS**

30515           None.

30516 **OPERANDS**

30517           The following operand is supported:

30518           *pid*       One of the following:

30519                       1. The unsigned decimal integer process ID of a command, for which the utility is to  
 30520                       wait for the termination.

30521                       2. A job control job ID (see the **XBD** specification, **Chapter 2, Glossary**) that  
 30522                       identifies a background process group to be waited for. The job control job ID  
 30523                       notation is applicable only for invocations of *wait* in the current shell execution  
 30524                       environment; see Section 2.12 on page 63. The exit status of *wait* is determined by  
 30525                       the last command in the pipeline.

30526                       **Note:** The job control job ID type of *pid* is available on systems supporting  
 30527                       both the job control option and the User Portability Utilities Option,  
 30528                       which applies to all XSI-conformant systems.

30529 **STDIN**

30530           Not used.

30531 **INPUT FILES**

30532           None.

30533 **ENVIRONMENT VARIABLES**30534           The following environment variables affect the execution of *wait*:

30535           **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
 30536           **LANG** is unset or null, the corresponding value from the implementation-dependent  
 30537           default locale will be used. If any of the internationalisation variables contains an  
 30538           invalid setting, the utility will behave as if none of the variables had been defined.

30539 **LC\_ALL**  
 30540 If set to a non-empty string value, override the values of all the other  
 30541 internationalisation variables.

30542 **LC\_CTYPE**  
 30543 Determine the locale for the interpretation of sequences of bytes of text data as  
 30544 characters (for example, single- as opposed to multi-byte characters in arguments).

30545 **LC\_MESSAGES**  
 30546 Determine the locale that should be used to affect the format and contents of diagnostic  
 30547 messages written to standard error.

30548 EX **NLSPATH**  
 30549 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

30550 **ASYNCHRONOUS EVENTS**  
 30551 Default.

30552 **STDOUT**  
 30553 Not used.

30554 **STDERR**  
 30555 Used only for diagnostic messages.

30556 **OUTPUT FILES**  
 30557 None.

30558 **EXTENDED DESCRIPTION**  
 30559 None.

30560 **EXIT STATUS**  
 30561 If one or more operands were specified, all of them have terminated or were not known by the  
 30562 invoking shell, and the status of the last operand specified is known, then the exit status of *wait*  
 30563 will be the exit status information of the command indicated by the last operand specified. If the  
 30564 process terminated abnormally due to the receipt of a signal, the exit status will be greater than  
 30565 128 and will be distinct from the exit status generated by other signals, but the exact value is  
 30566 unspecified. (See the *kill -l* option.) Otherwise, the *wait* utility will exit with one of the  
 30567 following values:

30568       0   The *wait* utility was invoked with no operands and all process IDs known by the  
 30569           invoking shell have terminated.

30570       1–126   The *wait* utility detected an error.

30571       127   The command identified by the last *pid* operand specified is unknown.

30572 **CONSEQUENCES OF ERRORS**  
 30573 Default.

30574 **APPLICATION USAGE**  
 30575 On most implementations, *wait* is a shell built-in. If it is called in a subshell or separate utility  
 30576 execution environment, such as one of the following:

30577       (*wait*)  
 30578       nohup *wait* ...  
 30579       find . -exec *wait* ... \;

30580 it will return immediately because there will be no known process IDs to wait for in those  
 30581 environments.

30582 Historical implementations of interactive shells have discarded the exit status of terminated  
 30583 background processes before each shell prompt. Therefore, the status of background processes

30584 was usually lost unless it terminated while *wait* was waiting for it. This could be a serious  
 30585 problem when a job that was expected to run for a long time actually terminated quickly with a  
 30586 syntax or initialisation error because the exit status returned was usually zero if the requested  
 30587 process ID was not found. This specification requires the implementation to keep the status of  
 30588 terminated jobs available until the status is requested, so that scripts like:

```
30589 j1&
30590 p1=$!
30591 j2&
30592 wait $p1
30593 echo Job 1 exited with status $?
30594 wait $!
30595 echo Job 2 exited with status $?
```

30596 will work without losing status on any of the jobs. The shell is allowed to discard the status of  
 30597 any process that it determines the application cannot get the process ID from the shell. It is also  
 30598 required to remember only {CHILD\_MAX} number of processes in this way. Since the only way  
 30599 to get the process ID from the shell is by using the "!" shell parameter, the shell is allowed to  
 30600 discard the status of an asynchronous list if \$! was not referenced before another asynchronous  
 30601 list was started. (This means that the shell only has to keep the status of the last asynchronous  
 30602 list started if the application did not reference \$!. If the implementation of the shell is smart  
 30603 enough to determine that a reference to \$! was not saved anywhere that the application can  
 30604 retrieve it later, it can use this information to trim the list of saved information. Note also that a  
 30605 successful call to *wait* with no operands discards the exit status of all asynchronous lists.)

30606 If the exit status of *wait* is greater than 128, there is no way for the application to know if the  
 30607 waited-for process exited with that value or was killed by a signal. Since most utilities exit with  
 30608 small values, there is seldom any ambiguity. Even in the ambiguous cases, most applications  
 30609 just need to know that the asynchronous job failed; it does not matter whether it detected an  
 30610 error and failed or was killed and did not complete its job normally.

#### 30611 EXAMPLES

30612 Although the exact value used when a process is terminated by a signal is unspecified, if it is  
 30613 known that a signal terminated a process, a script can still reliably figure out which signal using  
 30614 *kill* as shown by the following script:

```
30615 sleep 1000&
30616 pid=$!
30617 kill -kill $pid
30618 wait $pid
30619 echo $pid was terminated by a SIG$(kill -l $?) signal.
```

30620 If the following sequence of commands is run in less than 31 seconds:

```
30621 sleep 257 | sleep 31 &
30622 jobs -l %%
```

30623 either of the following commands will return the exit status of the second *sleep* in the pipeline:

```
30624 wait <pid of sleep 31>
30625 wait %%
```

#### 30626 FUTURE DIRECTIONS

30627 None.

#### 30628 SEE ALSO

30629 *sh*, the XSH specification description of *waitpid()*.

30630 **CHANGE HISTORY**

30631 First released in Issue 2.

30632 **Issue 4**

30633 Aligned with the ISO/IEC 9945-2: 1993 standard.

30634 **NAME**

30635       wc — word, line and byte or character count |

30636 **SYNOPSIS**30637       wc [-c|-m][*-lw*][*file...*] |30638 **DESCRIPTION**30639       The *wc* utility reads one or more input files and, by default, writes the number of newline  
30640       characters, words and bytes contained in each input file to the standard output.

30641       The utility also writes a total count for all named files, if more than one input file is specified.

30642       The *wc* utility considers a *word* to be a non-zero-length string of characters delimited by white  
30643       space.30644 **OPTIONS**30645       The *wc* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

30646       The following options are supported:

30647       **-c**       Write to the standard output the number of bytes in each input file.30648       **-l**       Write to the standard output the number of newline characters in each input file.30649       **-m**       Write to the standard output the number of characters in each input file.30650       **-w**       Write to the standard output the number of words in each input file.30651       When any option is specified, *wc* will report only the information requested by the specified  
30652       options.30653 **OPERANDS**

30654       The following operand is supported:

30655       *file*       A pathname of an input file. If no *file* operands are specified, the standard input will be  
30656       used.30657 **STDIN**30658       The standard input will be used only if no *file* operands are specified. See the INPUT FILES |  
30659       section.30660 **INPUT FILES**

30661       The input files may be of any type.

30662 **ENVIRONMENT VARIABLES**30663       The following environment variables affect the execution of *wc*:30664       **LANG**   Provide a default value for the internationalisation variables that are unset or null. If  
30665       **LANG** is unset or null, the corresponding value from the implementation-dependent |  
30666       default locale will be used. If any of the internationalisation variables contains an  
30667       invalid setting, the utility will behave as if none of the variables had been defined.30668       **LC\_ALL**30669       If set to a non-empty string value, override the values of all the other  
30670       internationalisation variables.30671       **LC\_CTYPE**30672       Determine the locale for the interpretation of sequences of bytes of text data as  
30673       characters (for example, single- as opposed to multi-byte characters in arguments and  
30674       input files) and which characters are defined as white space characters.



30675 **LC\_MESSAGES**

30676 Determine the locale that should be used to affect the format and contents of diagnostic  
 30677 messages written to standard error and informative messages written to standard  
 30678 output.

30679 EX **NLSPATH**

30680 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

30681 **ASYNCHRONOUS EVENTS**

30682 Default.

30683 **STDOUT**

30684 By default, the standard output contains an entry for each input file of the form:

30685 "%d %d %d %s\n", *<newlines>*, *<words>*, *<bytes>*, *<file>*

30686 If the **-m** option is specified, the number of characters replace the *<bytes>* field in this format.

30687 If any options are specified and the **-l** option is not specified, the number of newline characters  
 30688 will not be written.

30689 If any options are specified and the **-w** option is not specified, the number of words will not be  
 30690 written.

30691 If any options are specified and neither **-c** nor **-m** is specified, the number of bytes or characters  
 30692 will not be written.

30693 If no input *file* operands are specified, no name will be written and no blank characters  
 30694 preceding the pathname will be written.

30695 If more than one input *file* operand is specified, an additional line will be written, of the same  
 30696 format as the other lines, except that the word **total** (in the POSIX locale) will be written instead  
 30697 of a pathname and the total of each column will be written as appropriate. Such an additional  
 30698 line, if any, will be written at the end of the output.

30699 **STDERR**

30700 Used only for diagnostic messages.

30701 **OUTPUT FILES**

30702 None.

30703 **EXTENDED DESCRIPTION**

30704 None.

30705 **EXIT STATUS**

30706 The following exit values are returned:

30707 0 Successful completion.

30708 >0 An error occurred.

30709 **CONSEQUENCES OF ERRORS**

30710 Default.

30711 **APPLICATION USAGE**

30712 The **-m** option is not a switch, but an option at the same level as **-c**. Thus, to produce the full  
 30713 default output with character counts instead of bytes, the command required is:

30714 wc -mlw

30715 **EXAMPLES**

30716 None.

30717 **FUTURE DIRECTIONS**

30718 None.

30719 **SEE ALSO**30720 *cksum*.30721 **CHANGE HISTORY**

30722 First released in Issue 2.

30723 **Issue 4**

30724 Aligned with the ISO/IEC 9945-2: 1993 standard.

30725 **NAME**30726           what — identify SCCS files (**DEVELOPMENT**)30727 **SYNOPSIS**30728 EX        what [-s] *file...*30729 **DESCRIPTION**

30730       The *what* utility searches the given files for all occurrences of the pattern that *get* (see *get*)  
 30731       substitutes for %Z% (@(#)) and writes to standard output what follows until the first  
 30732       occurrence of one of the following:

30733               "     >    newline    \     NUL

30734 **OPTIONS**

30735       The *what* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**. The  
 30736       following option is supported:

30737        -s       Quit after finding the first occurrence of the pattern in each file.

30738 **OPERANDS**

30739       The following operands are supported:

30740       *file*     A pathname of a file to search.

30741 **STDIN**

30742       Not used.

30743 **INPUT FILES**

30744       The input files are of any file type.

30745 **ENVIRONMENT VARIABLES**

30746       The following environment variables affect the execution of *what*:

30747       *LANG*    Provide a default value for the internationalisation variables that are unset or null. If  
 30748       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 30749       default locale will be used. If any of the internationalisation variables contains an  
 30750       invalid setting, the utility will behave as if none of the variables had been defined.

30751       *LC\_ALL*

30752               If set to a non-empty string value, override the values of all the other  
 30753       internationalisation variables.

30754       *LC\_CTYPE*

30755               Determine the locale for the interpretation of sequences of bytes of text data as  
 30756       characters (for example, single- as opposed to multi-byte characters in arguments and  
 30757       input files).

30758       *LC\_MESSAGES*

30759               Determine the locale that should be used to affect the format and contents of diagnostic  
 30760       messages written to standard error.

30761       *NLSPATH*

30762               Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

30763 **ASYNCHRONOUS EVENTS**

30764       Default.

30765 **STDOUT**

30766           The standard output consists of the following for each *file* operand:

30767           " %s:\n\t%s\n", <pathname>, <identification string>

30768 **STDERR**

30769           Used only for diagnostic messages.

30770 **OUTPUT FILES**

30771           None.

30772 **EXTENDED DESCRIPTION**

30773           None.

30774 **EXIT STATUS**

30775           The following exit values are returned:

30776           0   Any matches were found.

30777           1   Otherwise.

30778 **CONSEQUENCES OF ERRORS**

30779           Default.

30780 **APPLICATION USAGE**

30781           The *what* utility is intended to be used in conjunction with the SCCS command *get*, which  
 30782           automatically inserts identifying information, but it can also be used where the information is  
 30783           inserted by any other means.

30784           When the string "@(#)" is included in a library routine in a shared library, it might not be found  
 30785           in an **a.out** file using that library routine.

30786 **EXAMPLES**

30787           If the C-language program in file **f.c** contains:

30788           char ident[] = "@(#)identification information";

30789           and **f.c** is compiled to yield **f.o** and **a.out**, then the command:

30790           what f.c f.o a.out

30791           will write:

30792           f.c:

30793           identification information

30794           ...

30795           f.o:

30796           identification information

30797           ...

30798           a.out:

30799           identification information

30800           ...

30801 **FUTURE DIRECTIONS**

30802           None.

30803 **SEE ALSO**

30804           *get*.

30805 **CHANGE HISTORY**

30806 First released in Issue 2.

30807 **Issue 4**

30808 Format reorganised.

30809 Utility Syntax Guidelines support mandated.

30810 Internationalised environment variable support mandated.

## 30811 NAME

30812       who — display who is on the system

## 30813 SYNOPSIS

30814 EX       who [-mu] [-s [-bHlprt]] [*file*]

30815 EX       who [-mTu] [-abdHlprt]] [*file*]

30816 EX       who -q [*file*]

30817 EX       who am i

30818 EX       who am I

## 30819 DESCRIPTION

30820       The *who* utility lists various pieces of information about accessible users. The domain of  
30821       accessibility is implementation-dependent.

30822 EX       Based on the options given, *who* also can list the user's name, terminal line, login time, elapsed  
30823       time since activity occurred on the line and the process ID of the command interpreter for each  
30824       current system user.

## 30825 OPTIONS

30826       The *who* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

30827       The following options are supported. The metavariables, such as *<line>*, refer to fields described  
30828       in the STDOUT section.

30829 EX       **-a**       Process the implementation-dependent database or named file with the **-b**, **-d**, **-l**, **-p**,  
30830       **-r**, **-t**, **-T** and **-u** options turned on.

30831 EX       **-b**       Write the time and date of the last reboot.

30832 EX       **-d**       Write a list of all processes that have expired and not been respawned by the *init*  
30833       system process. The *<exit>* field appears for dead processes and contains the  
30834       termination and exit values of the dead process. This can be useful in determining why  
30835       a process terminated.

30836 EX       **-H**       Write column headings above the regular output.

30837 EX       **-l**       (The letter ell.) List only those lines on which the system is waiting for someone to  
30838       login. The *<name>* field is **LOGIN** in such cases. Other fields are the same as for user  
30839       entries except that the *<state>* field does not exist.

30840       **-m**       Output only information about the current terminal.

30841 EX       **-p**       List any other process that is currently active and has been previously spawned by *init*.

30842 EX       **-q**       (Quick.) List only the names and the number of users currently logged on. When this  
30843       option is used, all other options are ignored.

30844 EX       **-r**       Write the current *run-level* of the *init* process.

30845 EX       **-s**       List only the *<name>*, *<line>* and *<time>* fields. This is the default case.

30846 EX       **-t**       Indicate the last change to the system clock.

30847       **-T**       Show the state of each terminal, as described in the STDOUT section.

30848 EX       **-u**       This option lists only those users who are currently logged in. Output the user's "idle  
30849       time" in addition to any other information. The idle time is the time since any activity  
30850       occurred on the user's terminal. The method of determining this is unspecified. The  
30851       *<name>* is the user's login name. The *<line>* is the name of the line as found in the

30852 directory **/dev**. The *<time>* is the time that the user logged in. The *<activity>* is the  
 30853 number of hours and minutes since activity last occurred on that particular line. A dot  
 30854 indicates that the terminal has seen activity in the last minute and is therefore  
 30855 “current.” If more than twenty-four hours have elapsed or the line has not been used  
 30856 since boot time, the entry is marked *<old>*. This field is useful when trying to  
 30857 determine whether a person is working at the terminal or not. The *<pid>* is the process  
 30858 ID of the user’s login process.

### 30859 OPERANDS

30860 EX The following operands are supported:

30861 **am i**  
 30862 **am I** In the POSIX locale, limit the output to describing the invoking user, equivalent to the  
 30863 **-m** option. The **am** and **i** or **I** must be separate arguments.

30864 **file** Specify a pathname of a file to substitute for the implementation-dependent database  
 30865 of logged-on users that *who* uses by default.

### 30866 STDIN

30867 Not used.

### 30868 INPUT FILES

30869 None.

### 30870 ENVIRONMENT VARIABLES

30871 The following environment variables affect the execution of *who*:

30872 **LANG** Provide a default value for the internationalisation variables that are unset or null. If  
 30873 **LANG** is unset or null, the corresponding value from the implementation-dependent  
 30874 default locale will be used. If any of the internationalisation variables contains an  
 30875 invalid setting, the utility will behave as if none of the variables had been defined.

30876 **LC\_ALL**  
 30877 If set to a non-empty string value, override the values of all the other  
 30878 internationalisation variables.

30879 **LC\_CTYPE**  
 30880 Determine the locale for the interpretation of sequences of bytes of text data as  
 30881 characters (for example, single- as opposed to multi-byte characters in arguments).

30882 **LC\_MESSAGES**  
 30883 Determine the locale that should be used to affect the format and contents of diagnostic  
 30884 messages written to standard error.

30885 **LC\_TIME**  
 30886 Determine the locale used for the format and contents of the date and time strings.

30887 EX **NLSPATH**  
 30888 Determine the location of message catalogues for the processing of **LC\_MESSAGES**.

### 30889 ASYNCHRONOUS EVENTS

30890 Default.

### 30891 STDOUT

30892 EX OF The *who* utility writes its default information to the standard output in the following general  
 30893 format:

30894 *<name>* [*<state>*] *<line>* *<time>* [*<activity>*] [*<pid>*] [*<comment>*] [*<exit>*]

30895

30896 The following format is used for the **-T** option:

30897       "%s %c %s %s\n", <name>, <terminal state>, <terminal name>,  
30898       <time of login>

30899 where <terminal state> is one of the following characters:

- 30900 + The terminal allows write access to other users.
- 30901 - The terminal denies write access to other users.
- 30902 ? The terminal write-access state cannot be determined.

30903 In the POSIX locale, the <time of login> is equivalent in format to the output of:

30904       date +"%b %e %H:%M"

30905 If the **-u** option is used with **-T**, the idle time is added to the end of the previous format in an  
30906 unspecified format.

#### 30907 **STDERR**

30908 Used only for diagnostic messages.

#### 30909 **OUTPUT FILES**

30910 None.

#### 30911 **EXTENDED DESCRIPTION**

30912 None.

#### 30913 **EXIT STATUS**

30914 The following exit values are returned:

- 30915       0 Successful completion.
- 30916       >0 An error occurred.

#### 30917 **CONSEQUENCES OF ERRORS**

30918 Default.

#### 30919 **APPLICATION USAGE**

30920 The name *init* used for the system process is the most commonly used on historical systems, but  
30921 it may vary.

30922 The “domain of accessibility” referred to is a broad concept that permits interpretation either on  
30923 a very secure basis or even to allow a network-wide implementation like the historical *rwho*.

#### 30924 **EXAMPLES**

30925 None.

#### 30926 **FUTURE DIRECTIONS**

30927 None.

#### 30928 **SEE ALSO**

30929 *msg*.



30930 **CHANGE HISTORY**

30931 First released in Issue 2.

30932 **Issue 4**

30933 Aligned with the ISO/IEC 9945-2: 1993 standard.

30934 **NAME**

30935       write — write to another user

30936 **SYNOPSIS**30937       write *user\_name* [*terminal*]30938 **DESCRIPTION**30939       The *write* utility reads lines from the user's standard input and writes them to the terminal of  
30940 another user. When first invoked, it writes the message:30941       **Message from** *sender-login-id* (*sending-terminal*) [*date*]...30942       to *user\_name*. When it has successfully completed the connection, the sender's terminal will be  
30943 alerted twice to indicate that what the sender is typing is being written to the recipient's  
30944 terminal.

30945       If the recipient wants to reply, this can be accomplished by typing:

30946       write *sender-login-id* [*sending-terminal*]30947       upon receipt of the initial message. Whenever a line of input as delimited by a NL, EOF or EOL  
30948 special character (see the **XBD** specification, **Chapter 9, General Terminal Interface**) is  
30949 accumulated while in canonical input mode, the accumulated data will be written on the other  
30950 user's terminal. Characters are processed as follows:

- 30951       • Typing the alert character will write the alert character to the recipient's terminal.
- 30952       • Typing the erase and kill characters will affect the sender's terminal in the manner described  
30953       by the **termios** interface in the **XBD** specification, **Chapter 9, General Terminal Interface**.
- 30954       • Typing the interrupt or end-of-file characters will cause *write* to write an appropriate  
30955       message (EOT\n in the POSIX locale) to the recipient's terminal and exit.
- 30956       • Typing characters from LC\_CTYPE classifications **print** or **space** will cause those characters  
30957       to be sent to the recipient's terminal.
- 30958       • When and only when the *stty ixten* local mode is enabled, the existence and processing of  
30959       additional special control characters and multi-byte or single-byte functions is  
30960       implementation-dependent.
- 30961       • Typing other non-printable characters will cause implementation-dependent sequences of  
30962       printable characters to be written to the recipient's terminal.

30963       To write to a user who is logged in more than once, the *terminal* argument can be used to indicate  
30964       which terminal to write to; otherwise, the recipient's terminal is selected in an implementation-  
30965       dependent manner and an informational message will be written to the sender's standard  
30966       output, indicating which terminal was chosen.

30967       Permission to be a recipient of a *write* message can be denied or granted by use of the *mesg*  
30968       utility. However, a user's privilege may further constrain the domain of accessibility of other  
30969       users' terminals. The *write* utility will fail when the user lacks the appropriate privileges to  
30970       perform the requested action.

30971 **OPTIONS**

30972       None.

30973 **OPERANDS**

30974 The following operands are supported:

30975 *user\_name*

30976 Login name of the person to whom the message will be written. This operand must be  
30977 of the form returned by the *who* utility.

30978 *terminal*

30979 Terminal identification in the same format provided by the *who* utility.

30980 **STDIN**

30981 Lines to be copied to the recipient's terminal will be read from standard input.

30982 **INPUT FILES**

30983 None.

30984 **ENVIRONMENT VARIABLES**

30985 The following environment variables affect the execution of *write*:

30986 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
30987 *LANG* is unset or null, the corresponding value from the implementation-dependent  
30988 default locale will be used. If any of the internationalisation variables contains an  
30989 invalid setting, the utility will behave as if none of the variables had been defined.

30990 *LC\_ALL*

30991 If set to a non-empty string value, override the values of all the other  
30992 internationalisation variables.

30993 *LC\_CTYPE*

30994 Determine the locale for the interpretation of sequences of bytes of text data as  
30995 characters (for example, single- as opposed to multi-byte characters in arguments and  
30996 input files). If the recipient's locale does not use an *LC\_CTYPE* equivalent to the  
30997 sender's, the results are undefined.

30998 *LC\_MESSAGES*

30999 Determine the locale that should be used to affect the format and contents of diagnostic  
31000 messages written to standard error and informative messages written to standard  
31001 output.

31002 EX *NLSPATH*

31003 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

31004 **ASYNCHRONOUS EVENTS**

31005 If an interrupt signal is received, *write* will write an appropriate message on the recipient's  
31006 terminal and exit with a status of zero. It will take the standard action for all other signals.

31007 **STDOUT**

31008 An informational message will be written to standard output if a recipient is logged in more  
31009 than once.

31010 **STDERR**

31011 Used only for diagnostic messages.

31012 **OUTPUT FILES**

31013 The recipient's terminal will be used for output.

31014 **EXTENDED DESCRIPTION**

31015 None.

**31016 EXIT STATUS**

31017 The following exit values are returned:

31018 0 Successful completion.

31019 >0 The addressed user is not logged on or the addressed user denies permission.

**31020 CONSEQUENCES OF ERRORS**

31021 Default.

**31022 APPLICATION USAGE**

31023 The *talk* utility is considered by some users to be a more usable utility on full-screen terminals.

**31024 EXAMPLES**

31025 None.

**31026 FUTURE DIRECTIONS**

31027 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
31028 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
31029 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
31030 finalised.

**31031 SEE ALSO**

31032 *mesg*, *talk*, *who*, the **XBD** specification, **Chapter 9, General Terminal Interface**.

**31033 CHANGE HISTORY**

31034 First released in Issue 2.

**31035 Issue 4**

31036 Aligned with the ISO/IEC 9945-2: 1993 standard.

**31037 Issue 5**

31038 FUTURE DIRECTIONS section added.

## 31039 NAME

31040 xargs — construct argument lists and invoke utility

## 31041 SYNOPSIS

```
31042 EX OB xargs [-t][-p][-e[eofstr]][-E eofstr][-I replstr][-i[replstr]]
31043 [-L number][-l[number]][-n number][-x]][-s size][utility [argument...]]
```

## 31044 DESCRIPTION

31045 The *xargs* utility constructs a command line consisting of the *utility* and *argument* operands  
 31046 specified followed by as many arguments read in sequence from standard input as will fit in  
 31047 length and number constraints specified by the options. The *xargs* utility then invokes the  
 31048 constructed command line and waits for its completion. This sequence is repeated until an end-  
 31049 of-file condition is detected on standard input or an invocation of a constructed command line  
 31050 returns an exit status of 255.

31051 Arguments in the standard input must be separated by unquoted blank characters, or unescaped  
 31052 blank characters or newline characters. A string of zero or more non-double-quote (") and non-  
 31053 newline characters can be quoted by enclosing them in double-quotes. A string of zero or more  
 31054 non-apostrophe (') and non-newline characters can be quoted by enclosing them in apostrophes.  
 31055 Any unquoted character can be escaped by preceding it with a backslash. The *utility* will be  
 31056 executed one or more times until the end-of-file is reached. The results are unspecified if the  
 31057 utility named by *utility* attempts to read from its standard input.

31058 The generated command line length will be the sum of the size in bytes of the utility name and  
 31059 each argument treated as strings, including a null byte terminator for each of these strings. The  
 31060 *xargs* utility will limit the command line length such that when the command line is invoked, the  
 31061 combined argument and environment lists (see the *exec* family of functions in the **XSH**  
 31062 specification) will not exceed {ARG\_MAX}–2048 bytes. Within this constraint, if neither the **–n**  
 31063 nor the **–s** option is specified, the default command line length will be at least {LINE\_MAX}.

## 31064 OPTIONS

31065 OB The *xargs* utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**, except  
 31066 that the **–e**, **–i** and **–I** take optional option-arguments that cannot be separate arguments.

31067 The following options are supported:

31068 EX OB **–e[*eofstr*]**  
 31069 Use *eofstr* as the logical end-of-file string. Underscore ( \_ ) is assumed for the logical  
 31070 EOF string if neither **–e** nor **–E** is used. When the **–eofstr** option-argument is omitted,  
 31071 the logical EOF string capability is disabled and underscores are taken literally. The  
 31072 *xargs* utility reads standard input until either end-of-file or the logical EOF string is  
 31073 encountered.

31074 EX **–E *eofstr***  
 31075 Specify a logical end-of-file string to replace the default underscore. The *xargs* utility  
 31076 reads standard input until either end-of-file or the logical EOF string is encountered.

31077 EX **–I *replstr***  
 31078 Insert mode: *utility* will be executed for each line from standard input, taking the entire  
 31079 line as a single argument, inserting it in *arguments* for each occurrence of *replstr*. A  
 31080 maximum of five arguments in *arguments* can each contain one or more instances of  
 31081 *replstr*. Any blank characters at the beginning of each line are ignored. Constructed  
 31082 arguments cannot grow larger than 255 bytes. Option **–x** is forced on. The **–I** and **–i**  
 31083 options are mutually exclusive; the last one specified takes effect.

|          |                     |                                                                                                             |
|----------|---------------------|-------------------------------------------------------------------------------------------------------------|
| 31084 OB | <b>-i[ replstr]</b> |                                                                                                             |
| 31085    |                     | This option is equivalent to <b>-I replstr</b> . The string {} is assumed for <i>replstr</i> if the option- |
| 31086    |                     | argument is omitted.                                                                                        |
| 31087 EX | <b>-L number</b>    |                                                                                                             |
| 31088    |                     | The <i>utility</i> will be executed for each non-empty <i>number</i> lines of arguments from                |
| 31089    |                     | standard input. The last invocation of <i>utility</i> will be with fewer lines of arguments if              |
| 31090    |                     | fewer than <i>number</i> remain. A line is considered to end with the first newline character               |
| 31091    |                     | unless the last character of the line is a blank character; a trailing blank character                      |
| 31092    |                     | signals continuation to the next non-empty line, inclusive. The <b>-L</b> , <b>-l</b> and <b>-n</b> options |
| 31093    |                     | are mutually exclusive; the last one specified takes effect.                                                |
| 31094 OB | <b>-l[ number]</b>  |                                                                                                             |
| 31095    |                     | (The letter ell.) This option is equivalent to <b>-L number</b> . If <i>number</i> is omitted, 1 is         |
| 31096    |                     | assumed. Option <b>-x</b> is forced on.                                                                     |
| 31097    | <b>-n number</b>    |                                                                                                             |
| 31098    |                     | Invoke <i>utility</i> using as many standard input arguments as possible, up to <i>number</i> (a            |
| 31099    |                     | positive decimal integer) arguments maximum. Fewer arguments will be used if:                               |
| 31100    |                     | • The command line length accumulated exceeds the size specified by the <b>-s</b> option                    |
| 31101    |                     | (or {LINE_MAX} if there is no <b>-s</b> option).                                                            |
| 31102    |                     | • The last iteration has fewer than <i>number</i> , but not zero, operands remaining.                       |
| 31103 EX | <b>-p</b>           | Prompt mode: the user is asked whether to execute <i>utility</i> at each invocation. Trace                  |
| 31104    |                     | mode ( <b>-t</b> ) is turned on to write the command instance to be executed, followed by a                 |
| 31105    |                     | prompt to standard error. An affirmative response read from <b>/dev/tty</b> will execute the                |
| 31106    |                     | command; otherwise, that particular invocation of <i>utility</i> is skipped.                                |
| 31107    | <b>-s size</b>      | Invoke <i>utility</i> using as many standard input arguments as possible yielding a command                 |
| 31108    |                     | line length less than <i>size</i> (a positive decimal integer) bytes. Fewer arguments will be               |
| 31109    |                     | used if:                                                                                                    |
| 31110    |                     | • The total number of arguments exceeds that specified by the <b>-n</b> option.                             |
| 31111 EX |                     | • The total number of lines exceeds that specified by the <b>-L</b> option.                                 |
| 31112    |                     | • End-of-file is encountered on standard input before <i>size</i> bytes are accumulated.                    |
| 31113    |                     | Values of <i>size</i> up to at least {LINE_MAX} bytes are supported, provided that the                      |
| 31114    |                     | constraints specified in the DESCRIPTION section are met. It is not considered an error                     |
| 31115    |                     | if a value larger than that supported by the implementation or exceeding the                                |
| 31116    |                     | constraints specified in the DESCRIPTION section is given; <i>xargs</i> will use the largest                |
| 31117    |                     | value it supports within the constraints.                                                                   |
| 31118    | <b>-t</b>           | Enable trace mode. Each generated command line will be written to standard error just                       |
| 31119    |                     | prior to invocation.                                                                                        |
| 31120 EX | <b>-x</b>           | Terminate if a command line containing <i>number</i> arguments (see the <b>-n</b> option above) or          |
| 31121    |                     | <i>number</i> lines (see the <b>-L</b> option above) will not fit in the implied or specified size (see     |
| 31122    |                     | the <b>-s</b> option above).                                                                                |

31123 **OPERANDS**

31124 The following operands are supported:

31125 *utility* The name of the utility to be invoked, found by search path using the *PATH*  
 31126 environment variable, described in the **XBD** specification, **Chapter 6, Environment**  
 31127 **Variables**. If *utility* is omitted, the default is the *echo* utility. If the *utility* operand  
 31128 names any of the special built-in utilities in Section 2.14 on page 67, the results are  
 31129 undefined.

31130 *argument*31131 An initial option or operand for the invocation of *utility*.31132 **STDIN**

31133 The standard input must be a text file. The results are unspecified if an end-of-file condition is  
 31134 detected immediately following an escaped newline character.

31135 **INPUT FILES**31136 EX The file */dev/tty* is used to read responses required by the **-p** option.31137 **ENVIRONMENT VARIABLES**31138 The following environment variables affect the execution of *xargs*:

31139 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 31140 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 31141 default locale will be used. If any of the internationalisation variables contains an  
 31142 invalid setting, the utility will behave as if none of the variables had been defined.

31143 *LC\_ALL*

31144 If set to a non-empty string value, override the values of all the other  
 31145 internationalisation variables.

31146 *LC\_COLLATE*

31147 Determine the locale for the behaviour of ranges, equivalence classes and multi-  
 31148 character collating elements used in the extended regular expression defined for the  
 31149 **yesexpr** locale keyword in the *LC\_MESSAGES* category.

31150 *LC\_CTYPE*

31151 Determine the locale for the interpretation of sequences of bytes of text data as  
 31152 characters (for example, single- as opposed to multi-byte characters in arguments and  
 31153 input files) and the behaviour of character classes used in the extended regular  
 31154 expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

31155 *LC\_MESSAGES*

31156 Determine the locale for the processing of affirmative responses and that should be  
 31157 used to affect the format and contents of diagnostic messages written to standard error.

31158 EX *NLSPATH*31159 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

31160 *PATH* Determine the location of *utility*, as described in the **XBD** specification, **Chapter 6,**  
 31161 **Environment Variables**.

31162 **ASYNCHRONOUS EVENTS**

31163 Default.

31164 **STDOUT**

31165 Not used.

31166 **STDERR**

31167 EX Used for diagnostic messages and the **-t** and **-p** options. If the **-t** option is specified, the *utility* and its constructed argument list will be written to standard error, as it will be invoked, prior to invocation. If **-p** is specified, a prompt of the following format will be written (in the POSIX locale):

31171 " ? . . . "

31172 at the end of the line of the output from **-t**.

31173 **OUTPUT FILES**

31174 None.

31175 **EXTENDED DESCRIPTION**

31176 None.

31177 **EXIT STATUS**

31178 The following exit values are returned:

- |       |       |                                                                                                                                                                                           |
|-------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31179 | 0     | All invocations of <i>utility</i> returned exit status zero.                                                                                                                              |
| 31180 | 1-125 | A command line meeting the specified requirements could not be assembled, one or more of the invocations of <i>utility</i> returned a non-zero exit status, or some other error occurred. |
| 31181 |       |                                                                                                                                                                                           |
| 31182 |       |                                                                                                                                                                                           |
| 31183 | 126   | The utility specified by <i>utility</i> was found but could not be invoked.                                                                                                               |
| 31184 | 127   | The utility specified by <i>utility</i> could not be found.                                                                                                                               |

31185 **CONSEQUENCES OF ERRORS**

31186 If a command line meeting the specified requirements cannot be assembled, the utility cannot be  
 31187 invoked, an invocation of the utility is terminated by a signal, or an invocation of the utility exits  
 31188 with exit status 255, the *xargs* utility will write a diagnostic message and exit without processing  
 31189 any remaining input.

31190 **APPLICATION USAGE**

31191 The 255 exit status (described as -1 in Issue 3) allows a utility being used by *xargs* to tell *xargs* to  
 31192 terminate if it knows no further invocations using the current data stream will succeed. Thus,  
 31193 *utility* should explicitly exit with an appropriate value to avoid accidentally returning with 255.

31194 Note that input is parsed as lines; blank characters separate arguments. If *xargs* is used to  
 31195 bundle output of commands like *find dir -print* or *ls* into commands to be executed, unexpected  
 31196 results are likely if any filenames contain any blank characters or newline characters. This can  
 31197 be fixed by using *find* to call a script that converts each file found into a quoted string that is then  
 31198 piped to *xargs*. Note that the quoting rules used by *xargs* are not the same as in the shell. They  
 31199 were not made consistent here because existing applications depend on the current rules and the  
 31200 shell syntax is not fully compatible with it. An easy rule that can be used to transform any string  
 31201 into a quoted form that *xargs* will interpret correctly is to precede each character in the string  
 31202 with a backslash.

31203 On implementations with a large value for {ARG\_MAX}, *xargs* may produce command lines  
 31204 longer than {LINE\_MAX}. For invocation of utilities, this is not a problem. If *xargs* is being used  
 31205 to create a text file, users should explicitly set the maximum command line length with the **-s**  
 31206 option.

31207 The *command*, *env*, *nice*, *nohup*, *time* and *xargs* utilities have been specified to use exit code 127 if  
 31208 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
 31209 utility exited with an error indication.” The value 127 was chosen because it is not commonly  
 31210 used for other meanings; most utilities use small values for “normal error conditions” and the  
 31211 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 31212 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some



31213 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 31214 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 31215 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
 31216 any other reason.

### 31217 EXAMPLES

31218 EX 1. The following will move all files from directory \$1 to directory \$2, and echo each move  
 31219 command just before doing it:

```
31220 ls $1 | xargs -I {} -t mv $1/{} $2/{} |
```

31221 2. The following command will combine the output of the parenthesised commands onto one  
 31222 line, which is then written to the end-of-file **log**:

```
31223 (logname; date; printf "%s\n" "$0 $*") | xargs >>log
```

31224 3. The following command will invoke *diff* with successive pairs of arguments originally  
 31225 typed as command line arguments (assuming there are no embedded blank characters in  
 31226 the elements of the original argument list):

```
31227 printf "%s\n" "$*" | xargs -n 2 -x diff
```

31228 EX 4. The user is asked which files in the current directory are to be archived. The files are  
 31229 archived into **arch**; a, one at a time, or b, many at a time.

```
31230 a. ls | xargs -p -L 1 ar -r arch
```

```
31231 b. ls | xargs -p -L 1 | xargs ar -r arch
```

31232 5. The following will execute with successive pairs of arguments originally typed as  
 31233 command line arguments:

```
31234 echo $* | xargs -n 2 diff
```

### 31235 FUTURE DIRECTIONS

31236 A version supporting the Utility Syntax Guidelines may be introduced.

31237 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 31238 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 31239 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 31240 finalised.

### 31241 SEE ALSO

31242 *echo*.

### 31243 CHANGE HISTORY

31244 First released in Issue 2.

#### 31245 Issue 4

31246 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 31247 Issue 5

31248 Second FUTURE DIRECTION added.

31249 **NAME**

31250 yacc — yet another compiler compiler (**DEVELOPMENT**)

31251 **SYNOPSIS**

31252 yacc [-dltv][*-b file\_prefix*][*-p sym\_prefix*] *grammar*

31253 **DESCRIPTION**

31254 The yacc utility reads a description of a context-free grammar in *file* and writes C source code,  
 31255 conforming to the ISO C standard, to a code file, and optionally header information into a  
 31256 header file, in the current directory. The C code defines a function and related routines and  
 31257 macros for an automaton that executes a parsing algorithm meeting the requirements in  
 31258 **Algorithms** on page 862.

31259 The form and meaning of the grammar are described in the EXTENDED DESCRIPTION section.

31260 The C source code and header file are produced in a form suitable as input for the C compiler  
 31261 (see *c89*).

31262 **OPTIONS**

31263 The yacc utility supports the **XBD** specification, **Section 10.2, Utility Syntax Guidelines**.

31264 The following options are supported:

31265 **-b file\_prefix**

31266 Use *file\_prefix* instead of *y* as the prefix for all output filenames. The code file **y.tab.c**,  
 31267 the header file **y.tab.h** (created when **-d** is specified), and the description file **y.output**  
 31268 (created when **-v** is specified), will be changed to *file\_prefix.tab.c*, *file\_prefix.tab.h*, and  
 31269 *file\_prefix.output*, respectively.

31270 **-d** Write the header file; by default only the code file is written. The **#define** statements  
 31271 that associate the token codes assigned by yacc with the user-declared token names.  
 31272 This allows source files other than **y.tab.c** to access the token codes.

31273 **-l** Produce a code file that does not contain any **#line** constructs. If this option is not  
 31274 present, it is unspecified whether the code file or header file contains **#line** directives.  
 31275 This should only be used after the grammar and the associated actions are fully  
 31276 debugged.

31277 **-p sym\_prefix**

31278 Use *sym\_prefix* instead of *yy* as the prefix for all external names produced by yacc. The  
 31279 names affected include the functions *yyparse()*, *yylex()* and *yyerror()*, and the variables  
 31280 *yyval*, *yychar* and *yydebug*. (In the remainder of this section, the six symbols cited are  
 31281 referenced using their default names only as a notational convenience.) Local names  
 31282 may also be affected by the **-p** option; however, the **-p** option does not affect **#define**  
 31283 symbols generated by yacc.

31284 **-t** Modify conditional compilation directives to permit compilation of debugging code in  
 31285 the code file. Run-time debugging statements will be always contained in the code file,  
 31286 but by default conditional compilation directives prevent their compilation.

31287 **-v** Write a file containing a description of the parser and a report of conflicts generated by  
 31288 ambiguities in the grammar.

**31289 OPERANDS**

31290 The following operand is required:

31291 *grammar*

31292 A pathname of a file containing instructions, hereafter called *grammar*, for which a  
 31293 parser is to be created. The format for the grammar is described in the EXTENDED  
 31294 DESCRIPTION section.

**31295 STDIN**

31296 Not used.

**31297 INPUT FILES**

31298 The file *grammar* must be a text file formatted as specified in the EXTENDED DESCRIPTION  
 31299 section.

**31300 ENVIRONMENT VARIABLES**

31301 The following environment variables affect the execution of *yacc*:

31302 *LANG* Provide a default value for the internationalisation variables that are unset or null. If  
 31303 *LANG* is unset or null, the corresponding value from the implementation-dependent  
 31304 default locale will be used. If any of the internationalisation variables contains an  
 31305 invalid setting, the utility will behave as if none of the variables had been defined.

31306 *LC\_ALL*

31307 If set to a non-empty string value, override the values of all the other  
 31308 internationalisation variables.

31309 *LC\_CTYPE*

31310 Determine the locale for the interpretation of sequences of bytes of text data as  
 31311 characters (for example, single- as opposed to multi-byte characters in arguments and  
 31312 input files).

31313 *LC\_MESSAGES*

31314 Determine the locale that should be used to affect the format and contents of diagnostic  
 31315 messages written to standard error.

31316 EX *NLSPATH*

31317 Determine the location of message catalogues for the processing of *LC\_MESSAGES*.

31318 The *LANG* and *LC\_\** variables affect the execution of the *yacc* utility as stated. The *main()*  
 31319 function defined in **Yacc Library** on page 861 calls:

31320 `setlocale(LC_ALL, "")`

31321 and thus, the program generated by *yacc* will also be affected by the contents of these variables  
 31322 at runtime.

**31323 ASYNCHRONOUS EVENTS**

31324 Default.

**31325 STDOUT**

31326 Not used.

**31327 STDERR**

31328 If shift/reduce or reduce/reduce conflicts are detected in *grammar*, *yacc* writes a report of those  
 31329 conflicts to the standard error in an unspecified format.

31330 Standard error is also used for diagnostic messages.

31331 **OUTPUT FILES**

31332       The code file, the header file and the description file are text files. All are described in the  
31333       following sections.

31334       **Code file**

31335       This file will contain the C source code for the *yyparse()* routine. It will contain code for the  
31336       various semantic actions with macro substitution performed on them as described in the  
31337       EXTENDED DESCRIPTION section. It will also contain a copy of the **#define** statements in the  
31338       header file. If a **%union** declaration is used, the declaration for YYSTYPE also will be included  
31339       in this file.

31340       The contents of the Program Section (see **Programs Section** on page 856) of the input file will  
31341       then be included.

31342       **Header file**

31343       The header file will contain **#define** statements that associate the token numbers with the token  
31344       names. This allows source files other than the code file to access the token codes. If a **%union**  
31345       declaration is used, the declaration for YYSTYPE and an extern YYSTYPE yylval declaration also  
31346       will be included in this file.

31347       **Description file**

31348       The description file will be a text file containing a description of the state machine  
31349       corresponding to the parser, using an unspecified format. Limits for internal tables (see **Limits**  
31350       in the EXTENDED DESCRIPTION section) will also be reported, in an implementation-  
31351       dependent manner. (Some implementations may use dynamic allocation techniques and have  
31352       no specific limit values to report.)

31353 **EXTENDED DESCRIPTION**

31354       The *yacc* command accepts a language that is used to define a grammar for a target language to  
31355       be parsed by the tables and code generated by *yacc*. The language accepted by *yacc* as a  
31356       grammar for the target language is described below using the *yacc* input language itself.

31357       The input *grammar* includes rules describing the input structure of the target language and code  
31358       to be invoked when these rules are recognised to provide the associated semantic action. The  
31359       code to be executed will appear as bodies of text that are intended to be C-language code. The  
31360       C-language inclusions are presumed to form a correct function when processed by *yacc* into its  
31361       output files. The code included in this way will be executed during the recognition of the target  
31362       language.

31363       Given a grammar, the *yacc* utility generates the files described in the OUTPUT FILES section.  
31364       The code file can be compiled and linked using *cc* or *c89*. If the declaration and programs  
31365       sections of the grammar file did not include definitions of *main()*, *yylex()* and *yyerror()*, the  
31366       compiled output requires linking with externally supplied version of those functions. Default  
31367       versions of *main()* and *yyerror()* are supplied in the *yacc* library and can be linked in by using the  
31368       **-ly** operand to *cc* or *c89*. The *yacc* library interfaces need not support interfaces with other than  
31369       the default **yy** symbol prefix. The application provides the lexical analyser function, *yylex()*; the  
31370       *lex* utility is specifically designed to generate such a routine.

### 31371 **Input Language**

31372 Every specification file must consist of three sections in order: *declarations*, *grammar rules* and  
 31373 *programs*, separated by double percent signs (%%). The declarations and programs sections can  
 31374 be empty. If the latter is empty, the preceding %% mark separating it from the rules section can  
 31375 be omitted.

31376 The input is free form text following the structure of the grammar defined below.

### 31377 **Lexical Structure of the Grammar**

31378 The characters blank, newline and form-feed are ignored, except that they must not appear in  
 31379 names or multi-character reserved symbols. Comments must be enclosed in /\* ... \*/, and can  
 31380 appear wherever a name is valid.

31381 Names are of arbitrary length, made up of letters, periods (.), underscores (\_) and non-initial  
 31382 digits. Upper- and lower-case letters are distinct. Portable applications must not use names  
 31383 beginning in yy or YY since the yacc parser uses such names. Many of the names appear in the  
 31384 final output of yacc, and thus they should be chosen to conform with any additional rules  
 31385 created by the C compiler to be used. In particular they will appear in **#define** statements.

31386 A literal consists of a single character enclosed in single-quotes (''). All of the escape sequences  
 31387 supported for character constants by the ISO C standard are supported by yacc.

31388 The relationship with the lexical analyser is discussed in detail below.

31389 The NUL character must not be used in grammar rules or literals.

### 31390 **Declarations Section**

31391 The declarations section is used to define the symbols used to define the target language and  
 31392 their relationship with each other. In particular, much of the additional information required to  
 31393 resolve ambiguities in the context-free grammar for the target language is provided here.

31394 Usually yacc assigns the relationship between the symbolic names it generates and their  
 31395 underlying numeric value. The declarations section makes it possible to control the assignment  
 31396 of these values.

31397 It is also possible to keep semantic information associated with the tokens currently on the parse  
 31398 stack in a user-defined C-language **union**, if the members of the union are associated with the  
 31399 various names in the grammar. The declarations section provides for this as well.

31400 The first group of declarators below all take a list of names as arguments. That list can  
 31401 optionally be preceded by the name of a C union member (called a *tag* below) appearing within  
 31402 "<" and ">". (As an exception to the typographical conventions of the rest of this specification, in  
 31403 this case <tag> does not represent a metavariable, but the literal angle bracket characters  
 31404 surrounding a symbol.) The use of *tag* specifies that the tokens named on this line are to be of  
 31405 the same C type as the union member referenced by *tag*. This is discussed in more detail below.

31406 For lists used to define tokens, the first appearance of a given token can be followed by a  
 31407 positive integer (as a string of decimal digits). If this is done, the underlying value assigned to it  
 31408 for lexical purposes will be taken to be that number.

31409 %token [*<tag>*] *name* [*number*] [*name* [*number*]]...

31410 Declares *names* to be a token. If *tag* is present, the C type for all tokens on this line will  
 31411 be declared to be the type referenced by *tag*. If a positive integer, *number*, follows a  
 31412 *name*, that value will be assigned to the token.

```

31413 %left [<tag>] name [number] [name [number]]...
31414 %right [<tag>] name [number] [name [number]]...
31415 Declares name to be a token, and assigns precedence to it. One or more lines, each
31416 beginning with one of these symbols, can appear in this section. All tokens on the same
31417 line have the same precedence level and associativity; the lines are in order of
31418 increasing precedence or binding strength. %left denotes that the operators on that
31419 line are left associative, and %right similarly denotes right associative operators. If tag
31420 is present, it declares a C type for names as described for %token.

31421 %nonassoc [<tag>] name [number] [name [number]]...
31422 Declares name to be a token, and indicates that this cannot be used associatively. If the
31423 parser encounters associative use of this token it will report an error. If tag is present, it
31424 declares a C type for names as described for %token.

31425 %type [<tag>] name...
31426 Declares that union member names are non-terminals, and thus it is required to have a
31427 tag field at its beginning. Because it deals with non-terminals only, assigning a token
31428 number or using a literal is also prohibited. If this construct is present, yacc will
31429 perform type checking; if this construct is not present, the parse stack will hold only the
31430 int type.

31431 Every name used in grammar undefined by a %token, %left, %right or %nonassoc declaration is
31432 assumed to represent a non-terminal symbol. The yacc utility will report an error for any non-
31433 terminal symbol that does not appear on the left side of at least one grammar rule.

31434 Once the type, precedence or token number of a name is specified, it will not be changed. If the
31435 first declaration of a token does not assign a token number, yacc will assign a token number.
31436 Once this assignment is made, the token number will not be changed by explicit assignment.

31437 The following declarators do not follow the previous pattern.

31438 %start name
31439 Declares the non-terminal name to be the start symbol, which represents the largest,
31440 most general structure described by the grammar rules. By default, it is the left-hand
31441 side of the first grammar rule; this default can be overridden with this declaration.

31442 %union { body of union (in C) }
31443 Declares the yacc value stack to be a union of the various types of values desired. By
31444 default, the values returned by actions (see below) and the lexical analyser will be
31445 integers. The yacc utility keeps track of types, and will insert corresponding union
31446 member names in order to perform strict type checking of the resulting parser.

31447 Alternatively, given that at least one <tag> construct is used, the union can be declared
31448 in a header file (which will be included in the declarations section by using an #include
31449 construct within %{ and %}), and a typedef used to define the symbol YYSTYPE to
31450 represent this union. The effect of %union is to provide the declaration of YYSTYPE
31451 directly from the input.

31452 %{ ... %}
31453 C-language declarations and definitions can appear in the declarations section,
31454 enclosed by these marks. These statements will be copied into the code file, and have
31455 global scope within it so that they can be used in the rules and program sections.

31456 The declarations section must be terminated by the token %%.

```

31457 **Grammar Rules in yacc**

31458 The rules section defines the context-free grammar to be accepted by the function *yacc* generates,  
 31459 and associates with those rules C-language actions and additional precedence information. The  
 31460 grammar is described below, and a formal definition follows.

31461 The rules section is comprised of one or more grammar rules. A grammar rule has the form:

31462       A : BODY ;

31463 The symbol A represents a non-terminal name, and **BODY** represents a sequence of zero or  
 31464 more *names*, *literals* and *semantic actions* that can then be followed by optional *precedence rules*.  
 31465 Only the names and literals participate in the formation of the grammar; the semantic actions  
 31466 and precedence rules are used in other ways. The colon and the semicolon are *yacc* punctuation.  
 31467 If there are several successive grammar rules with the same left-hand side, the vertical bar "|" |  
 31468 can be used to avoid rewriting the left-hand side; in this case the semicolon appears only after  
 31469 the last rule. The BODY part can be empty (or empty of names and literals) to indicate that the  
 31470 non-terminal symbol matches the empty string.

31471 The *yacc* utility assigns a unique number to each rule. Rules using the vertical bar notation are  
 31472 distinct rules. The number assigned to the rule appears in the description file.

31473 The elements comprising a BODY are:

31474       *name*

31475       *literal*   These form the rules of the grammar: *name* is either a *token* or a *non-terminal*; *literal*  
 31476                   stands for itself (less the lexically required quotation marks).

31477       *semantic action*

31478               With each grammar rule, the user can associate actions to be performed each time the  
 31479               rule is recognised in the input process. (Note that the word "action" can also refer to  
 31480               the actions of the parser—shift, reduce, and so on.) |

31481               These actions can return values and can obtain the values returned by previous actions.  
 31482               These values will be kept in objects of type YYSTYPE (see %**union**). The result value of  
 31483               the action will be kept on the parse stack with the left-hand side of the rule, to be  
 31484               accessed by other reductions as part of their right-hand side. By using the <tag>  
 31485               information provided in the declarations section, the code generated by *yacc* can be  
 31486               strictly type checked and contain arbitrary information. In addition, the lexical  
 31487               analyser can provide the same kinds of values for tokens, if desired.

31488               An action is an arbitrary C statement and as such can do input or output, call  
 31489               subprograms and alter external variables. An action is one or more C statements  
 31490               enclosed in curly braces "{" and "}". |

31491               Certain pseudo-variables can be used in the action. These are macros for access to data  
 31492               structures known internally to *yacc*.

31493       **\$\$**       The value of the action can be set by assigning it to \$\$ . If type checking is  
 31494                   enabled and the type of the value to be assigned cannot be determined, a  
 31495                   diagnostic message may be generated.

31496       **\$number** This refers to the value returned by the component specified by the token  
 31497                   *number* in the right side of a rule, reading from left to right; *number* can be zero  
 31498                   or negative. If it is, it refers to the data associated with the name on the  
 31499                   parser's stack preceding the leftmost symbol of the current rule. (That is, \$0  
 31500                   refers to the name immediately preceding the leftmost name in the current  
 31501                   rule, to be found on the parser's stack and \$-1 refers to the symbol to *its* left.)  
 31502                   If *number* refers to an element past the current point in the rule, or beyond the

31503 bottom of the stack, the result is undefined. If type checking is enabled and  
 31504 the type of the value to be assigned cannot be determined, a diagnostic  
 31505 message may be generated.

31506 **\$<tag>number**

31507 These correspond exactly to the corresponding symbols without the *tag*  
 31508 inclusion, but allow for strict type checking (and preclude unwanted type  
 31509 conversions). The effect is that the macro is expanded to use *tag* to select an  
 31510 element from the YYSTYPE union (using *dataname.tag*). This is particularly  
 31511 useful if *number* is not positive.

31512 **\$<tag>\$** This imposes on the reference the type of the union member referenced by *tag*.  
 31513 This construction is applicable when a reference to a left context value occurs  
 31514 in the grammar, and provides yacc with a means for selecting a type.

31515 Actions can occur in the middle of a rule as well as at the end; an action can access  
 31516 values returned by actions to its left, and in turn the value it returns can be accessed by  
 31517 actions to its right. An action appearing in the middle of a rule will be equivalent to  
 31518 replacing the action with a new non-terminal symbol and adding an empty rule with  
 31519 that non-terminal symbol on the left-hand side. The semantic action associated with  
 31520 the new rule will be equivalent to the original action. The use of actions within rules  
 31521 might introduce conflicts that would not otherwise exist.

31522 By default, the value of a rule is the value of the first element in it. If the first element  
 31523 does not have a type (particularly in the case of a literal) and type checking is turned on  
 31524 by **%type** an error message will result.

31525 *precedence*

31526 The keyword **%prec** can be used to change the precedence level associated with a  
 31527 particular grammar rule. Examples of this are in cases where a unary and binary  
 31528 operator have the same symbolic representation, but need to be given different  
 31529 precedences, or where the handling of an ambiguous if-else construction is necessary.  
 31530 The reserved symbol **%prec** can appear immediately after the body of the grammar  
 31531 rule and can be followed by a token name or a literal. It will cause the precedence of  
 31532 the grammar rule to become that of the following token name or literal. The action for  
 31533 the rule as a whole can follow **%prec**.

31534 If a program section follows, the grammar rules must be terminated by **%%**.

### 31535 **Programs Section**

31536 The *programs* section can include the definition of the lexical analyser *yylex()*, and any other  
 31537 functions, for example those used in the actions specified in the grammar rules. This is C-  
 31538 language code, and will be included in the code file after the tables and code generated by yacc.  
 31539 It is unspecified whether the programs section precedes or follows the semantic actions in the  
 31540 output file; therefore, if the application contains any macro definitions and declarations intended  
 31541 to apply to the code in the semantic actions, it must place them within **{...%}** in the  
 31542 declarations section.



```

31543 Input Grammar
31544 The following input to yacc yields a parser for the input to yacc. This formal syntax takes
31545 precedence over the preceding text syntax description.
31546 The lexical structure is defined less precisely; Lexical Structure of the Grammar on page 853
31547 defines most terms. The correspondence between the previous terms and the tokens below is as
31548 follows.
31549 IDENTIFIER
31550 This corresponds to the concept of name, given previously. It also includes literals as
31551 defined previously.
31552 C_IDENTIFIER
31553 This is a name, and additionally it is known to be followed by a colon. A literal cannot
31554 yield this token.
31555 NUMBER
31556 A string of digits (a non-negative decimal integer).
31557 TYPE
31558 LEFT
31559 MARK
31560 and so on
31561 These correspond directly to %type, %left, %% and so on.
31562 { ... } This indicates C-language source code, with the possible inclusion of "$" macros as
31563 discussed previously.
31564 /* Grammar for the input to yacc */
31565 /* Basic entries */
31566 /* The following are recognised by the lexical analyser */
31567 %token IDENTIFIER /* includes identifiers and literals */
31568 %token C_IDENTIFIER /* identifier (but not literal)
31569 followed by a : */
31570 %token NUMBER /* [0-9][0-9]* */
31571 /* Reserved words : %type=>TYPE %left=>LEFT, and so on */
31572 %token LEFT RIGHT NONASSOC TOKEN PREC TYPE START UNION
31573 %token MARK /* the %% mark */
31574 %token LCURL /* the %{ mark */
31575 %token RCURL /* the }% mark */
31576 /* 8-bit character literals stand for themselves; */
31577 /* tokens have to be defined for multi-byte characters */
31578 %start spec
31579 %%
31580 spec : defs MARK rules tail
31581 ;
31582 tail : MARK
31583 {
31584 /* In this action, set up the rest of the file */
31585 }
31586 | /* empty; the second MARK is optional */

```

```

31587 ;
31588 defs : /* empty */
31589 | defs def
31590 ;
31591 def : START IDENTIFIER
31592 | UNION
31593 {
31594 /* Copy union definition to output */
31595 }
31596 | LCURL
31597 {
31598 /* Copy C code to output file */
31599 }
31600 | RCURL
31601 | rword tag nlist
31602 ;
31603 rword : TOKEN
31604 | LEFT
31605 | RIGHT
31606 | NONASSOC
31607 | TYPE
31608 ;
31609 tag : /* empty: union tag id optional */
31610 | '<' IDENTIFIER '>'
31611 ;
31612 nlist : nmno
31613 | nlist nmno
31614 ;
31615 nmno : IDENTIFIER /* Note: literal invalid with % type */
31616 | IDENTIFIER NUMBER /* Note: invalid with % type */
31617 ;
31618 /* rule section */
31619 rules : C_IDENTIFIER rbody prec
31620 | rules rule
31621 ;
31622 rule : C_IDENTIFIER rbody prec
31623 | '|' rbody prec
31624 ;
31625 rbody : /* empty */
31626 | rbody IDENTIFIER
31627 | rbody act
31628 ;
31629 act : '{'
31630 {
31631 /* Copy action, translate $$, and so on */
31632 }
31633 | '}'
31634 ;
31635 prec : /* empty */
31636 | PREC IDENTIFIER
31637 | PREC IDENTIFIER act

```

```

31638 | prec ';'
31639 ;

```

## 31640 Conflicts

31641 The parser produced for an input grammar may contain states in which conflicts occur. The  
 31642 conflicts occur because the grammar is not LALR(1). An ambiguous grammar always contains  
 31643 at least one LALR(1) conflict. The *yacc* utility will resolve all conflicts, using either default rules  
 31644 or user-specified precedence rules.

31645 Conflicts are either shift/reduce conflicts or reduce/reduce conflicts. A shift/reduce conflict is  
 31646 where, for a given state and lookahead symbol, both a shift action and a reduce action are  
 31647 possible. A reduce/reduce conflict is where, for a given state and lookahead symbol, reductions  
 31648 by two different rules are possible.

31649 The rules below describe how to specify what actions to take when a conflict occurs. Not all  
 31650 shift/reduce conflicts can be successfully resolved this way because the conflict may be due to  
 31651 something other than ambiguity, so incautious use of these facilities can cause the language  
 31652 accepted by the parser to be much different from that which was intended. The description file  
 31653 will contain sufficient information to understand the cause of the conflict. Where ambiguity is  
 31654 the reason either the default or explicit rules should be adequate to produce a working parser.

31655 The declared precedences and associativities (see **Declarations Section** on page 853) are used to  
 31656 resolve parsing conflicts as follows:

- 31657 1. A precedence and associativity is associated with each grammar rule; it is the precedence  
 31658 and associativity of the last token or literal in the body of the rule. If the **%prec** keyword is  
 31659 used, it overrides this default. Some grammar rules might not have both precedence and  
 31660 associativity.
- 31661 2. If there is a shift/reduce conflict, and both the grammar rule and the input symbol have  
 31662 precedence and associativity associated with them, then the conflict is resolved in favour  
 31663 of the action (shift or reduce) associated with the higher precedence. If the precedences are  
 31664 the same, then the associativity is used; left associative implies reduce, right associative  
 31665 implies shift, and non-associative implies an error in the string being parsed.
- 31666 3. When there is a shift/reduce conflict that cannot be resolved by rule 2, the shift is done.  
 31667 Conflicts resolved this way are counted in the diagnostic output described in **Error**  
 31668 **Handling**.
- 31669 4. When there is a reduce/reduce conflict, a reduction is done by the grammar rule that  
 31670 occurs earlier in the input sequence. Conflicts resolved this way are counted in the  
 31671 diagnostic output described in **Error Handling**.

31672 Conflicts resolved by precedence or associativity will not be counted in the shift/reduce and  
 31673 reduce/reduce conflicts reported by *yacc* on either standard error or in the description file.

## 31674 Error Handling

31675 The token **error** is reserved for error handling. The name **error** can be used in grammar rules. It  
 31676 indicates places where the parser can recover from a syntax error. The default value of **error** is  
 31677 256. Its value can be changed using a **%token** declaration. The lexical analyser should not  
 31678 return the value of **error**. (Multi-byte characters should be recognised by the lexical analyser and  
 31679 returned as tokens. They should not be returned as multi-byte character literals. The token **error**  
 31680 that is used for error recovery is normally assigned the value 256 in the historical  
 31681 implementation. Thus, the token value 256, which used in many multi-byte character sets, is not  
 31682 available for use as the value of a user-defined token.)

31683 The parser will detect a syntax error when it is in a state where the action associated with the  
31684 lookahead symbol is **error**. A semantic action can cause the parser to initiate error handling by  
31685 executing the macro YYERROR. When YYERROR is executed, the semantic action will pass  
31686 control back to the parser. YYERROR cannot be used outside of semantic actions.

31687 When the parser detects a syntax error, it normally calls **yyerror** with the character string  
31688 "syntax error" as its argument. The call will not be made if the parser is still recovering from a  
31689 previous error when the error is detected. The parser is considered to be recovering from a  
31690 previous error until the parser has shifted over at least three normal input symbols since the last  
31691 error was detected or a semantic action has executed the macro **yyerrok**. The parser will not call  
31692 **yyerror** when YYERROR is executed.

31693 The macro function YYRECOVERING() will return 1 if a syntax error has been detected and the  
31694 parser has not yet fully recovered from it. Otherwise, zero will be returned.

31695 When a syntax error is detected by the parser, the parser will check if a previous syntax error has  
31696 been detected. If a previous error was detected, and if no normal input symbols have been  
31697 shifted since the preceding error was detected, the parser checks if the lookahead symbol is an  
31698 endmarker (see **Interface to the Lexical Analyser**). If it is, the parser will return with a non-zero  
31699 value. Otherwise, the lookahead symbol will be discarded and normal parsing will resume.

31700 When YYERROR is executed or when the parser detects a syntax error and no previous error has  
31701 been detected, or at least one normal input symbol has been shifted since the previous error was  
31702 detected, the parser will pop back one state at a time until the parse stack is empty or the current  
31703 state allows a shift over **error**. If the parser empties the parse stack, it will return with a non-zero  
31704 value. Otherwise, it will shift over **error** and then resume normal parsing. If the parser reads a  
31705 lookahead symbol before the error was detected, that symbol will still be the lookahead symbol  
31706 when parsing is resumed.

31707 The macro **yyerrok** in a semantic action will cause the parser to act as if it has fully recovered  
31708 from any previous errors. The macro **yyclearin** will cause the parser to discard the current  
31709 lookahead token. If the current lookahead token has not yet been read, **yyclearin** will have no  
31710 effect.

31711 The macro YYACCEPT will cause the parser to return with the value zero. The macro  
31712 YYABORT will cause the parser to return with a non-zero value.

### 31713 **Interface to the Lexical Analyser**

31714 The **yylex()** function is an integer-valued function that returns a *token number* representing the  
31715 kind of token read. If there is a value associated with the token returned by **yylex()** (see the  
31716 discussion of *tag* above), it will be assigned to the external variable **yyval**.

31717 If the parser and **yylex()** do not agree on these token numbers, reliable communication between  
31718 them cannot occur. For (one character) literals, the token is simply the numeric value of the  
31719 character in the current character set. The numbers for other tokens can either be chosen by **yacc**,  
31720 or chosen by the user. In either case, the **#define** construct of C is used to allow **yylex()** to return  
31721 these numbers symbolically. The **#define** statements are put into the code file, and the header  
31722 file if that file is requested. The set of characters permitted by **yacc** in an identifier is larger than  
31723 that permitted by C. Token names found to contain such characters will not be included in the  
31724 **#define** declarations.

31725 If the token numbers are chosen by **yacc**, the tokens other than literals will be assigned numbers  
31726 greater than 256, although no order is implied. A token can be explicitly assigned a number by  
31727 following its first appearance in the declarations section with a number. Names and literals not  
31728 defined this way retain their default definition. All assigned token numbers will be unique and  
31729 distinct from the token numbers used for literals. If duplicate token numbers cause conflicts in

31730 parser generation, *yacc* will report an error; otherwise, it is unspecified whether the token  
 31731 assignment is accepted or an error is reported.

31732 The end of the input is marked by a special token called the *endmarker*, which has a token  
 31733 number that is zero or negative. (These values are invalid for any other token.) All lexical  
 31734 analysers will return zero or negative as a token number upon reaching the end of their input. If  
 31735 the tokens up to, but excluding, the endmarker form a structure that matches the start symbol,  
 31736 the parser will accept the input. If the endmarker is seen in any other context, it will be  
 31737 considered an error.

### 31738 **Completing the Program**

31739 In addition to *yparse()* and *yylex()*, the functions *yyerror()* and *main()* are required to make a  
 31740 complete program. The application can supply *main()* and *yyerror()*, or those routines can be  
 31741 obtained from the *yacc* library.

### 31742 **Yacc Library**

31743 The following functions appear only in the *yacc* library accessible through the *-ly* operand to *cc*  
 31744 or *c89*; they can therefore be redefined by a portable application:

#### 31745 **int main(void)**

31746 This function will call *yparse()* and exit with an unspecified value. Other actions within  
 31747 this function are unspecified.

#### 31748 **int yyerror(const char \*s)**

31749 This function will write the NUL-terminated argument to standard error, followed by a  
 31750 newline character.

31751 The order of the *-ly* and *-ll* operands given to *cc* or *c89* is significant; the application must  
 31752 either provide its own *main()* function or ensure that *-ly* precedes *-ll*.

### 31753 **Debugging the Parser**

31754 The parser generated by *yacc* will have diagnostic facilities in it that can be optionally enabled at  
 31755 either compile time or at run time (if enabled at compile time). The compilation of the runtime  
 31756 debugging code is under the control of *YYDEBUG*, a preprocessor symbol. If *YYDEBUG* has a  
 31757 non-zero value, the debugging code will be included. If its value is zero, the code will not be  
 31758 included.

31759 In parsers where the debugging code has been included, the external *int yydebug* can be used to  
 31760 turn debugging on (with a non-zero value) and off (zero value) at run time. The initial value of  
 31761 *yydebug* will be zero.

31762 When *-t* is specified, the code file will be built such that, if *YYDEBUG* is not already defined at  
 31763 compilation time (using the *c89 -D YYDEBUG* option, for example), *YYDEBUG* will be set  
 31764 explicitly to 1. When *-t* is not specified, the code file will be built such that, if *YYDEBUG* is not  
 31765 already defined, it will be set explicitly to zero.

31766 The format of the debugging output is unspecified but includes at least enough information to  
 31767 determine the shift and reduce actions, and the input symbols. It also provides information  
 31768 about error recovery.

**Algorithms**

The parser constructed by **yacc** implements an LALR(1) parsing algorithm as documented in the literature. It is unspecified whether the parser is table-driven or direct-coded.

A parser generated by **yacc** will never request an input symbol from `yylex()` while in a state where the only actions other than the error action are reductions by a single rule.

The literature of parsing theory defines these concepts.

**Limits**

The **yacc** utility may have several internal tables. The minimum maximums for these tables are shown in the following table. The exact meaning of these values is implementation-dependent. The implementation will define the relationship between these values and between them and any error messages that the implementation may generate should it run out of space for any internal structure. An implementation may combine groups of these resources into a single pool as long as the total available to the user does not fall below the sum of the sizes specified by this section.

| Limit      | Minimum<br>Maximum | Description                                                                                                                                                                                                                                            |
|------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {NTERMS}   | 126                | Number of tokens.                                                                                                                                                                                                                                      |
| {NNONTERM} | 200                | Number of non-terminals.                                                                                                                                                                                                                               |
| {NPROD}    | 300                | Number of rules.                                                                                                                                                                                                                                       |
| {NSTATES}  | 600                | Number of states.                                                                                                                                                                                                                                      |
| {MEMSIZE}  | 5200               | Length of rules. The total length, in names (tokens and non-terminals), of all the rules of the grammar. The left-hand side is counted for each rule, even if it is not explicitly repeated, as specified in <b>Grammar Rules in yacc</b> on page 855. |
| {ACTSIZE}  | 4000               | Number of actions. “Actions” here (and in the description file) refer to parser actions (shift, reduce, and so on) not to semantic actions defined in <b>Grammar Rules in yacc</b> on page 855.                                                        |

**Table 3-17** Internal Limits in **yacc**

**EXIT STATUS**

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

If any errors are encountered, the run is aborted and **yacc** exits with a non-zero status. Partial code files and header files may be produced. The summary information in the description file will always be produced if the `-v` flag is present.

**APPLICATION USAGE**

Historical implementations experience name conflicts on the names `yacc.tmp`, `yacc.acts`, `yacc.debug`, `y.tab.c`, `y.tab.h` and `y.output` if more than one copy of **yacc** is running in a single directory at one time. The `-b` option was added to overcome this problem. The related problem of allowing multiple **yacc** parsers to be placed in the same file was addressed by adding a `-p` option to override the previously hard-coded `yy` variable prefix.

31814 The description of the **-p** option specifies the minimal set of function and variable names that  
 31815 cause conflict when multiple parsers are linked together. YYSTYPE does not need to be  
 31816 changed. Instead, the programmer can use **-b** to give the header files for different parsers  
 31817 different names, and then the file with the *yylex()* for a given parser can include the header for  
 31818 that parser. Names such as *yyclearerr* do not need to be changed because they are used only in  
 31819 the actions; they do not have linkage. It is possible that an implementation will have other  
 31820 names, either internal ones for implementing things such as *yyclearerr*, or providing non-  
 31821 standard features that it wants to change with **-p**.

31822 Unary operators that are the same token as a binary operator in general need their precedence  
 31823 adjusted. This is handled by the **%prec** advisory symbol associated with the particular grammar  
 31824 rule defining that unary operator. See **Grammar Rules in yacc** on page 855. Applications are  
 31825 not required to use this operator for unary operators, but the grammars that do not require it are  
 31826 rare.

### 31827 EXAMPLES

31828 Access to the *yacc* library is obtained with library search operands to *cc* or *c89*. To use the *yacc*  
 31829 library *main()*:

```
31830 c89 y.tab.c -l y
```

31831 Both the *lex* library and the *yacc* library contain *main()*. To access the *yacc main()*:

```
31832 c89 y.tab.c lex.yy.c -l y -l l
```

31833 This ensures that the *yacc* library is searched first, so that its *main()* is used.

31834 The historical *yacc* libraries have contained two simple functions that are normally coded by the  
 31835 application programmer. These library functions are similar to the following code:

```
31836 #include <locale.h>
31837 int main(void)
31838 {
31839 extern int yyparse();
31840
31841 setlocale(LC_ALL, "");
31842
31843 /* If the following parser is one created by lex, the
31844 application must be careful to ensure that LC_CTYPE
31845 and LC_COLLATE are set to the POSIX locale. */
31846 (void) yyparse();
31847 return (0);
31848 }
```

```
31847 #include <stdio.h>
```

```
31848 int yyerror(const char *msg)
31849 {
31850 (void) fprintf(stderr, "%s\n", msg);
31851 return (0);
31852 }
```

### 31853 FUTURE DIRECTIONS

31854 The IEEE PASC 1003.2 Interpretations Committee has forwarded concerns about parts of this  
 31855 interface definition to the IEEE PASC Shell and Utilities Working Group which is identifying the  
 31856 corrections. A future revision of this specification will align with IEEE Std. 1003.2b when  
 31857 finalised.

31858 **SEE ALSO**31859 *cc, c89, lex.*31860 **CHANGE HISTORY**

31861 First released in Issue 2. |

31862 **Issue 4**

31863 Aligned with the ISO/IEC 9945-2: 1993 standard. |

31864 **Issue 5**

31865 FUTURE DIRECTIONS section added. |



31866 **NAME**

31867           zcat — expand and concatenate data

31868 **SYNOPSIS**31869 EX       zcat [*file...*]31870 **DESCRIPTION**

31871       The *zcat* utility will write to standard output the uncompressed form of files that have been  
 31872       compressed using the *compress* utility. It is the equivalent of *uncompress -c*. Input files are not  
 31873       affected.

31874 **OPTIONS**

31875       None.

31876 **OPERANDS**

31877       The following operand is supported:

31878       *file*       The pathname of a file previously processed by the *compress* utility. If *file* already has  
 31879       the *.Z* suffix specified, it is used as submitted. Otherwise, the *.Z* suffix is appended to  
 31880       the filename prior to processing.

31881 **STDIN**31882       The standard input will be used only if no *file* operands are specified, or if a *file* operand is "-".31883 **INPUT FILES**31884       Input files must be compressed files that are in the format produced by the *compress* utility.31885 **ENVIRONMENT VARIABLES**31886       The following environment variables affect the execution of *zcat*:

31887       *LANG*       Provide a default value for the internationalisation variables that are unset or null. If  
 31888       *LANG* is unset or null, the corresponding value from the implementation-dependent  
 31889       default locale will be used. If any of the internationalisation variables contains an  
 31890       invalid setting, the utility will behave as if none of the variables had been defined.

31891       *LC\_ALL*

31892       If set to a non-empty string value, override the values of all the other  
 31893       internationalisation variables.

31894       *LC\_CTYPE*

31895       Determine the locale for the interpretation of sequences of bytes of text data as  
 31896       characters (for example, single- as opposed to multi-byte characters in arguments).

31897       *LC\_MESSAGES*

31898       Determine the locale that should be used to affect the format and contents of diagnostic  
 31899       messages written to standard error.

31900       *NLSPATH*31901       Determine the location of message catalogues for the processing of *LC\_MESSAGES*.31902 **ASYNCHRONOUS EVENTS**

31903       Default.

31904 **STDOUT**

31905       The compressed files given as input will be written on standard output in their uncompressed  
 31906       form.

31907 **STDERR**

31908       Used only for diagnostic messages.

**31909 OUTPUT FILES**

31910           None.

**31911 EXTENDED DESCRIPTION**

31912           None.

**31913 EXIT STATUS**

31914           The following exit values are returned:

31915           0   Successful completion.

31916           >0  An error occurred.

**31917 CONSEQUENCES OF ERRORS**

31918           Default.

**31919 APPLICATION USAGE**

31920           None.

**31921 EXAMPLES**

31922           None.

**31923 FUTURE DIRECTIONS**

31924           None.

**31925 SEE ALSO**

31926           *compress, uncompress, zcat.*

**31927 CHANGE HISTORY**

31928           First released in Issue 4.