UniFLEX™ Utility Commands

Technical Systems Consultants, Inc.

UniFLEX™ Utility Commands

COPYRIGHT INFORMATION

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

UniFLEX Utility Commands

The UniFLEX Operating System contains a wide variety of utility commands. Commands exist for general file manipulations such as creating files, deleting them, moving them about the system, and listing their contents. Other commands exist for creating and controlling tasks. There are also many commands for general system maintenance and upkeep. This document provides the complete description of all of the currently supplied system utilities. Some commands may also have additional information available in separate documents (e.g. the text editor).

asmb

This is the absolute 6809 assembler supplied with UniFLEX.

SYNTAX

```
asmb file ... [+options]
```

DESCRIPTION

The assembler accepts one or more input source files from the calling line and, according to options specified, produces an assembled source listing and/or a binary output file. There are basically two types of options. The first is single character options of which more than one may be supplied per plus sign:

```
+a    print absolute address on relative branches
+b    do not create binary output file
+d    do not print date in page header
+f    do not perform auto-fielding of source
+g    list all code generated by fcb fdb, and fcc
+l    suppress assembled listing output
+n    include decimal line numbers in listing
+s    suppress printing of sorted symbol table
+w    suppress warning messages
+0000 set number of symbols the symbol table accepts
```

Obviously the last option listed above is not a single character, but rather a decimal number. The second type of option is a single character, followed by an equals sign, followed by some parameter:

```
+a=prm   command line parameter a
+b=prm   command line parameter b
+c=prm   command line parameter c
+m=00    specify macro space in kilobytes (2 to 12)
+o=file  specify output binary file name
+p=000   specify starting page number for listing
```

Some valid examples follow:

```
asmb /usr/john/sample
asmb equates file* +bna
asmb dumptxt +o=dump +ls +2400
```

SEE ALSO

The full user's manual for the UniFLEX 6809 Assembler.

badblocks

Add the specified block to the file of bad blocks.

SYNTAX

/etc/badblocks <dev_name> <address_list> [+dimpqsv]

DESCRIPTION

The "badblocks" command adds the specified blocks to the file "/.badblocks" (also known as the bad-blocks file) on the specified device. The blocks in this file are inaccessible to the operating system. Thus, the user can remove a bad block from circulation by putting it in the bad-blocks file. When it has added all the specified blocks to the bad-blocks file, "badblocks" calls "diskrepair".

Arguments

<dev_name>	The name of the device which contains the bad blocks.
<address_list>	A list of the addresses (in decimal) of the blocks to place in the bad-blocks file.

Options Available

d	Do not run "diskrepair".
i	Ignore the restriction on repairing the disk in multi-user mode, and, when calling "diskrepair", use the ´i´ option. By default, "badblocks" does not allow the user to repair the root device (system disk) unless the system is in single-user mode. However, certain systems--such as the Tektronix 4404--cannot operate in single-user mode. In order to execute the "badblocks" command on such a system, the user must specify the ´i´ option. Under no circumstances should this option be used on a system which can operate in single-user mode.
m=<block_num>	If a block must be used for mapping, use the one specified by this option. By default, "badblocks" takes such a block from the free list. However, if structural problems or media errors make it impossible to do so, the ´m´ option can be used to tell "badblocks" which blocks to use for any necessary mapping. The ´m´ option may be used up to twelve times in one command, but it should never be

(continued)

necessary to use it more than twice.

p	Prompt for permission to make repairs while running "diskrepair".
q	Use quiet mode when running "diskrepair".
s	Ignore the validity of the system information record (SIR) when determining the size of the disk, the size of the fdn space, and the size of the swap space. This option should not be used unless the "badblocks" command fails due to problems in the SIR.
v	Run "diskrepair" in verbose mode.

EXAMPLES

1. /etc/badblocks /dev/w0 596 10321
2. /etc/badblocks /dev/w0 596 10321 +pv

The first example places block numbers 596 and 10321 into the file "/.badblocks" on the disk in drive w0. It then calls "diskrepair".

The second example places block numbers 596 and 10321 into the file "/.badblocks" on the disk in drive w0. It then calls "diskrepair" with the 'p' and 'v' options.

MESSAGES

Total bad blocks = <num_of_bad_blocks>
The "badblocks" command sends this message to standard output when it successfully completes its job.

ERROR MESSAGES

The error messages listed in this section are those messages which are returned by the "badblocks" command. A user who is running "badblocks" without the 'd' option may receive other error messages, which come from the "diskrepair" command and are explained in the documentation for "diskrepair".

Bad ".badblocks" file.
Either the file ".badblocks" is not a regular file or the first block of the file is located where the boot sector, the SIR, or the root directory should be. The user may be able to salvage information from the disk, but must eventually reformat it.

Can't call "diskrepair".
The "badblocks" command cannot read or execute the file "/etc/diskrepair".

(continued)

Can't determine mode.
By default, the "badblocks" command cannot repair the root device unless the system is in single-user mode (see the 'i' option). Therefore, if the device specified on the command line is the root device, "badblocks" must determine what mode the system is operating in. This error message means that "badblocks" cannot access the file "/act/utmp" (it is the length of this file that tells what mode is in effect).

Can't do badblocks on multiple devices.
The user must specify exactly one device on the command line.

Can't open device.
The operating system returned an error when "badblocks" tried to open the specified device or read its fdn.

Can't read existing ".badblocks"
The bad-blocks file is so badly damaged physically that "badblocks" cannot read it. The user may be able to salvage information from the disk, but must eventually reformat it.

Can't read System Information Record.
The SIR is so badly damaged physically that "diskrepair" cannot read it. The user may be able to salvage information from the disk, but must eventually reformat it.

Can't stat root.
The "badblocks" command cannot read the fdn which describes the root directory. The user may be able to salvage some information from the disk, but must eventually reformat it.

Can't stat std. output.
The "badblocks" command cannot read the fdn of whatever file is opened as standard output. The user should rerun "badblocks" with the terminal as standard output.

Data overlaps swap space.
The information in the SIR indicates that the data space overlaps the swap space. The user should execute "diskrepair" to fix this problem. If and only if "diskrepair" fails to fix the problem, the user should execute "badblocks" with the 's' option.

Device is busy.
The "badblocks" command must function when the disk is not in use. Therefore, "badblocks" determines whether or not the specified disk is mounted and tries to unmount a mounted disk before proceeding. This error message means that either some user's working directory is on the specified disk or some user is accessing a file on that disk.

(continued)

Disk too large or bad size in SIR.
The "badblocks" command checks to see that the size of the disk is within the range that it can handle. The current limit is approximately 400 Megabytes. If the data in the SIR indicate that the disk is larger than this limit, "badblocks" issues this error message. This error is also fatal to "diskrepair". The user should salvage as much information as possible and reformat the disk.

Error reading ".badblocks" file.
The "badblocks" command must read the file "/.badblocks" before it adds any blocks to the file. This message indicates that it encountered an I/O error when it tried to read the file.

Error reading block <block_num>.
The specified block contains an I/O error.

Error reading fdn <fdn_num> in block <block_num>.
The specified fdn contains an I/O error.

Error writing block <block_num>.
The operating system returned an I/O error when "badblocks" tried to write to the specified block.

Error writing fdn <fdn_number> in block <block_num>.
The operating system returned an I/O error when "badblocks" tried to write to the specified fdn.

Error writing new ".badblocks"
The operating system returned an I/O error when "badblocks" tried to write the new bad-blocks file. The user may be able to salvage information from the disk, but must eventually reformat it.

Ignoring block <block_num> -- Already in ".badblocks"
The specified block is already in the bad-blocks file.

Ignoring block <block_num> -- Too large
The specified address is larger than the number of blocks on the disk.

Ignoring block <block_number> -- Too small
The "badblocks" command cannot put a block whose address is less than 3 into the file "/.badblocks" because those blocks are essential to the proper functioning of the operating system.

Invalid argument to 'm' option
The argument to the 'm' option must be a decimal number.

Invalid option: '<char>'
The option specified by <char> is not a valid option to the "badblocks" command.

(continued)

Must be in single-user mode.
  Unless the 'i' option is in effect, "badblocks" cannot repair the root device unless the system is in single-user mode. The 'i' option should be used if and only if the system cannot operate in single-user mode (see the 'i' option).

No ".badblocks" file on "<dev_name>".
  In order for the "badblocks" command to succeed, the bad-blocks file must already exist. The user should salvage as much information as possible, reformat the disk, and verify that the "format" program created a bad-blocks file.

No device specified.
  The user did not specify a device on the command line.

No free blocks available for mapping.
  The "badblocks" command needed a block to use for mapping in extending the file "/.badblocks", but no blocks were available. The user can reexecute "badblocks" with the 'm' option. It is wise to specify as the argument to the 'm' option a block which is in the swap space.

No such device.
  The user specified a nonexistent device on the command line.

Not a block device.
  The "badblocks" command can only operate on a block device.

Output directed to specified device.
  When putting blocks in the bad-blocks file, it is impractical to try to redirect the output to the device that is being repaired. The user should reexecute "badblocks" without redirecting the output or redirecting it to a different, mounted device.

Permission denied.
  A user who executes the "badblocks" command must have both read and write permission on the specified device.

SIR fdn block count error: <size>
  The number of blocks reserved for fdns exceeds either 65,535 (the maximum allowed by UniFLEX) or the size of the disk. This error is also fatal to "diskrepair". The user should salvage as much information as possible and reformat the disk.

SIR pointer to free space is invalid.
  The "badblocks" command needs a block from the free space to use for mapping in extending the bad-blocks file; however, because an internal pointer to the free space is invalid, it cannot obtain the necessary block. The user should should reexecute "badblocks" using the 'm' option. It is wise to specify as the argument to the 'm' option a block which is in the swap space.

(continued)

Too many badblocks on "<dev_name>".
The maximum number of blocks the bad-blocks file can hold is 800.

Unmount error <error_number>
The "badblocks" command encountered some problem other than a busy device when it tried to unmount the device. The accompanying error number is the number of the UniFLEX error that caused the failure. The user should consult the operating system manual for an explanation of the error.

SEE ALSO

diskrepair
format

blockcheck

This command is used to check the integrity of all file blocks on the disk, as well as the list of free blocks.

SYNTAX

```
/etc/blockcheck block_device_name
```

DESCRIPTION

The 'block_device_name' is the name of a block device. Two examples will demonstrate its use.

```
/etc/blockcheck /dev/fd1
/etc/blockcheck /dev/hd0
```

The first example will check the floppy disk drive 1 and the second example will check the hard disk. The output from this command is a complete summary of disk blocks on the tested disk. The last section of the summary is called 'Error Summary' and contains the information regarding bad disks. If any of the error information lines contains a non-zero block count, the disk is bad and should not be used. The only exception to this is the 'missing blocks' count. If this is non-zero, there will be lost blocks but this is not a degenerating situation.

SEE ALSO

devcheck, fdncheck

chd

The chd command is used to change to a new working directory.

SYNTAX

```
chd [directory]
```

DESCRIPTION

The directory specified in the command will become the working or default directory. If no directory is specified, the login directory will be used. You must have execute permission in the directory specified or an error will result. This command is part of shell and does not reside on the disk. An example will demonstrate its use.

```
chd /usr/john
chd book
```

The first line will change the working directory to '/usr/john'. The second example will change to the directory 'book' which resides in the current working directory.

SEE ALSO

shell

check

Check will read an entire file or list of files and report any read errors encountered.

SYNTAX

```
check file ...
```

DESCRIPTION

Check will read each file specified and report any read errors encountered. If an error is reported, the file should be considered damaged and unusable. Several examples follow.

```
check test
check *
check /usr/john/file bargs
```

The first example will check the file 'test' in the working directory. The next example will check all of the files in the working directory. The last line will check the file 'file' in the directory '/usr/john' and the file 'bargs' in the working directory.

copy

Make a copy of a file or list of files. A new file is created which exactly duplicates the file being copied.

SYNTAX

```
copy file1 file2
copy file ... directory
```

DESCRIPTION

The first form of the copy command is used to make a copy of 'file1' and give it the name 'file2'. The second form of copy is used to copy a list of files into the specified directory. In this form, the original file names are kept. In all cases, only regular files may be copied (not directories). If the new file specification already exists, it will be deleted. The new file will retain the original's permissions and owner. If the file did not exist, the permissions will be that of the source file. Some examples follow:

```
copy parts parts.bak
copy letter /usr/john/letters
copy test1 test2 pow /usr/john
```

The first example will copy the file named 'parts' into the file named 'parts.bak'. If 'parts.bak' already existed, it will be deleted without notice! The second line will copy the file 'letter' in the working directory into the file named 'letters' in the directory '/usr/john'. The final example will copy the files 'test1', 'test2', and 'pow' into the directory '/usr/john'. Each of these files will retain its original name.

SEE ALSO

link, move, rename

crdir

This command is used to create new directories.

SYNTAX

```
crdir name ...
```

DESCRIPTION

Create a new directory for each name specified. The user must have write permission in the parent directory (the directory in which the new directory will be created). Each new directory will contain the standard entries '.' and '..' which represent the directory itself and its parent, respectively. Directories are given 'rwxrwx' permissions when created but may be altered by the default permissions. The permissions may of course be changed at anytime by using the 'perms' command. A few examples will demonstrate the use of crdir.

```
crdir book
crdir junk /usr/john/bin
```

The first line will create a new directory called 'book' in the working directory. The second example will create two new directories. The first one will be named 'junk' and will reside in the working directory and the second one will be named 'bin' and will reside in the directory '/usr/john'.

SEE ALSO

dperm, kill, perms

create

The create command is used to create an empty file.

SYNTAX

```
create file ...
```

DESCRIPTION

Create will create an empty file for each name specified. Some system mechanisms require an existing file before they will work. Mail is an example, it needs a file named '.mail' (see mail). The text editor could be used for this purpose but the create command is much simpler. As an example:

```
create .mail
```

will create an empty file named '.mail' in the current working directory.

SEE ALSO

mail

date

Date is used to display the current date and time. It can also be used to set these values.

SYNTAX

```
date [ [MM-DD[-YY]] HH:MM[:SS] ]
```

DESCRIPTION

The date command with no arguments is used to display the local current date and time. Only the system manager may set the time. The 'MM-DD-YY' argument represents month-day-year. If this argument is omitted, the current values will remain. The 'HH:MM:SS' argument sets the hours:minutes:seconds part of the time. If the seconds are omitted from the specification they will be set to zero. The time entered should be the local time. The system keeps the internal time in UTC (GMT) but performs all of the necessary conversions for time zone and Daylight Savings. A few examples will demonstrate the use of date.

```
date 3-20-80 16:32:44
date 8:30
date 6-12 12:5
```

The first example will set the local time and date to March 20, 1980, 4:32:44 PM (time is kept in a 24 hour clock). The second line will set the time to 8:30 AM but leave the day, month, and year to their current values. The last example will set the date to June 12 (the current year is unchanged) and sets the time to 12:05.

SEE ALSO

history

devcheck

This command is used to check a disk for CRC errors.

SYNTAX

```
/etc/devcheck block_device_name
```

DESCRIPTION

The 'block_device_name' is the name of a block device. An example of this commands use is:

```
/etc/devcheck /dev/fd0
```

This command will check the floppy drive 0 for CRC errors. If no errors are found, no output is generated by the command. The command quits as soon as one error is detected. If any errors are found, the disk should be considered bad and not used. It is a good idea to run this command immediately after formatting a disk to check it for bad spots. Note that this command is ineffective on hard disks which have bad tracks even though the tracks have been removed by the formatting program.

SEE ALSO

blockcheck, fdncheck

devcheck

Check a device for I/O errors.

SYNTAX

/etc/devcheck <dev_name_list> [+bdDfsv]

DESCRIPTION

The "devcheck" command checks the specified device for I/O errors. As it checks the device, it prints informative messages, which tell the user what part of the device is being checked. It always checks the boot sector and the system information record (SIR). By default, it also checks the fdn space, the swap space, and the volume space. Every time it finds a bad block, it prints a message giving the address of the block in both decimal and hexadecimal. When it is finished, "devcheck" prints a message reporting the total number of bad blocks on the disk.

If a floppy disk contains one or more bad blocks, it should probably be discarded. If a hard disk contains one or more bad blocks, these should be removed by either running the "badblocks" program, rerunning "devcheck" with the ´b´ option, or reformatting the disk with the addresses of all bad blocks placed in the file "/.badblocks" (also known as the bad-blocks file). It is wise to run this command immediately after formatting a disk.

Arguments

<dev_name_list>	A list of the names of the devices to check. They must be block devices.

Options Available

b	Put all bad blocks in the file "/.badblocks".
d	Do not call "diskrepair" after putting bad blocks in "/.badblocks". This option requires the ´b´ option and is incompatible with the ´D´ option.
D=<opt_list>	Specifies a list of options to use when calling "diskrepair" after putting bad blocks in the bad-blocks file. This option requires the ´b´ option and is incompatible with the ´d´ option.
f	Check only the fdn space.
s	Check only the swap space.
v	Check only the volume space.

(continued)

EXAMPLES

1. /etc/devcheck /dev/fd0
2. /etc/devcheck /dev/fd0 +v
3. /etc/devcheck /dev/fd0 +bD=v

The first example checks the entire disk in floppy drive 0 for I/O errors.

The second example checks the boot sector, the SIR, and the volume space of the device in floppy drive 0 for I/O errors.

The third example checks the entire disk in floppy drive 0 for I/O errors. All bad blocks detected are put into the bad-blocks file. If it puts any blocks into the bad-blocks file, "devcheck" calls "diskrepair" with the 'v' option.

MESSAGES

Badblocks file too large - continuing without list.
"Devcheck" cannot read a bad-blocks file that has more than 138 bad blocks in it. Currently, this theoretical limitation on the number of bad blocks is unlikely to present a practical limitation. The number of bad blocks on a disk should not even approach 138.

Can't read '.badblocks' file - continuing without list.
The "devcheck" command encountered an I/O error when it tried to read the file ".badblocks".

File '.badblocks' not found - continuing with check.
The device specified does not contain a file named ".badblocks", or due to damage in the logical structure of the disk, "devcheck" cannot locate the file.

ERROR MESSAGES

"<dev_name>" is not a block device.
The device specified is not a block device.

Cannot execute "badblocks".
The "devcheck" command attempted to call "badblocks" to put a bad block into the bad-blocks file. However, the "badblocks" program could not be executed. Most probably the "badblocks" program does not exist on the system disk or the user does not have permission to execute it.

Cannot execute "diskrepair".
The "devcheck" command attempted to call "diskrepair" after having put bad blocks into the bad-blocks file. However, the "diskrepair"

(continued)

program could not be executed. Most probably the "diskrepair" program does not exist on the system disk or the user does not have permission to execute it.

Error opening "<dev_name>" : <reason>
The operating system returned an error when "devcheck" tried to open the specified device. This message is followed by an interpretation of the error returned by the operating system.

Incompatible options: ´d´ and ´D´.
The ´d´ and ´D´ options are incompatible. The ´D´ option is ignored.

Invalid option ´<char>´: ignored.
The option specified by <char> is not a valid option to the "devcheck" command.

No ".badblocks" file, ´b´ option ignored.
A ".badblocks" file must exist on the disk being tested in order for the ´b´ option to be valid.

Syntax: /etc/devcheck <dev_name_list> [+bdDfsv]
The "devcheck" command expects at least one argument. This message indicates that the argument count is wrong.

The ´<char>´ option requires the ´b´ option.
The option specified by <char> requires that the user also specify the ´b´ option. The option specified by <char> is ignored.

SEE ALSO

badblocks
diskrepair
format

dir

The dir command is used to list the contents of directories.

SYNTAX

```
dir [+ablrst] [directory] ...
```

DESCRIPTION

The dir command will list the contents of the specified directories. If no directories are named as command arguments, the contents of the current working directory will be displayed. Normally the file names which reside in the directory will be listed by name only, alphabetically sorted, and several names per display line. This display format may be altered with the several possible options. The options are as follows:

+a	display all files (those starting with '.' also)
+l	long listing (see below)
+b	display length of file in bytes (long listing only)
+r	reverse the sense of the sort
+s	display file names one per line (short listing only)
+t	sort files by last modified time (most recent first)

Dir will normally not display file names which start with a period ('.'). The +a option will allow those files to be displayed as well. The +l option provides great detail about each file name in the directory. The files are listed one per line with the file name appearing first on the line. Following the name is the file size in blocks (or bytes if the +b option was specified). The next field specifies the file type as defined by the following:

b	block special file (device)
c	character special file (device)
d	directory
blank	regular type file

The majority of files will be regular files and therefore have this field blank. The following field is the link count and specifies how many directory entries are linked to this file.

The next field contains the permissions associated with the file. The permission field is made up of six columns, the first three represent the user's permissions (owner), the second three represent all others' permissions. The columns are in the order 'rwx' which are described below.

r	read permission granted
w	write permission granted
x	execute permission granted
-	the permission for this field is denied

The next field on the line is the file owner. An owner's name is the same as his login name. Finally, the last information provided is the time and date of the last modification made to the file. An example line from a long dir listing might look like this:

```
uniflex        32   1 rwxr-x  system  12:04 Jul 08 1980
```

This line shows the file 'uniflex' which is 32 blocks long (the +b option was not used), is a regular file, has a link count of 1, the owner can read, write, and execute it but others can only read and execute it (no write permission granted), its owner is 'system', and the last time it was modified was 12:04 on July 8, 1980. A few example calling lines for dir follow:

```
dir
dir +lt
dir +l +t
dir / /usr +l
```

The first line will display all file names in the current working directory (with the exception of those starting with '.'). The next two examples will give a long listing of the working directory with the files sorted by modification time. The last line will display the contents of the directory '/' (the root) and the directory '/usr'. Both listings will be in the long format. Note that the options may be anywhere on the command line.

diskrepair

Check and, optionally, repair inconsistencies in the logical structure of a disk.

SYNTAX

/etc/diskrepair <dev_name_list> [+abfimmpqruvz]

Arguments

<dev_name_list> A list of the names of the devices to check.

Options Available

- a Automatically place in the file "/.badblocks" any bad blocks encountered, and continue to run "diskrepair" until the disk is fixed.
- b Perform "blockcheck" only.
- f Perform "fdncheck" only.
- i Ignore the restriction on repairing the disk in multi-user mode. This option should be used if and only if the system cannot operate in single-user mode.
- m Ignore missing blocks.
- n Do not attempt to fix errors.
- p Prompt for permission to repair.
- q Use quiet mode.
- r Rebuild the free list whether or not it is in error.
- u Report on the usage of disk blocks.
- v Use verbose mode.
- z Do not print messages about possible errors in the sizes of files.

DESCRIPTION

The "diskrepair" utility checks the structure of the disk or disks specified in <dev_name_list>. The structure of the disk refers to the layout of and the connections among files, directories, free space, swap space, and other information that makes up the file system. Any inconsistencies in the structure are reported and, optionally, repaired. Although "diskrepair" does not methodically search for and repair media (I/O) errors, it can take care of any bad blocks it encounters. If the 'a' option is in effect when "diskrepair" encounters an I/O error, it calls the utility "/etc/badblocks", which places the offending block in the file "/.badblocks".

While it is operating, "diskrepair" calls two other utilities--"blockcheck" and "fdncheck", which are both located in the directory "/etc". "Blockcheck" is concerned with the allocation of blocks on the

(continued)

disk. It locates problems such as duplicate blocks, missing blocks, and invalid block addresses. "Fdncheck" is concerned with the directories on the disk. It locates problems such as unreferenced files, file names with invalid associated files, and so forth.

There are two major modes of operation, simple and verbose. The simple mode is selected by default; the verbose mode is selected by the 'v' option. In the verbose mode "diskrepair" reports all detected errors. In the simple mode it reports only those errors which require the deletion of files or of directory entries. If executed in simple mode, "diskrepair" issues a message upon completion which informs the user whether or not the disk is in need of repair.

By default, all detected errors are automatically repaired (if possible). Two options exist to alter the handling of errors. The 'n' option instructs "diskrepair" not to repair any errors. The 'p' option instructs "diskrepair" to prompt the user for permission to repair the errors it reports. In verbose mode this option causes "diskrepair" to prompt the user regarding all errors. In the simple mode, the user is prompted only for those errors which require the deletion of files or of directory entries; all other errors are automatically repaired without prompting. It should be noted that most repairs result in a loss of data. The user can generally infer which data have been lost from the messages displayed.

When using the command in simple mode (without the 'v' option), the user need not understand what types of checks are made by "diskrepair". The only decisions required are whether or not to delete the reported files. In verbose mode, much more information is given to the user. While this document is not intended to give full details of this information, the following list shows most of the inconsistencies in disk structure for which "diskrepair" checks. First, however, a few definitions are in order. A "file descriptor node" (or fdn) is an area on the disk which contains all the information the system needs about a file. There should always be one fdn per file on the disk. A UniFLEX directory entry is simply a file name and a pointer to the proper fdn. There may be multiple directory entries pointing to the same fdn (multiple names for the same file). Each pointer to an fdn is called a "link" to that file. If there is a file with no links, it is considered "unreferenced". "Out-of-range" refers to a pointer to a disk block or to an fdn which is beyond the valid number of blocks or fdns for the disk being tested. Here now, is a partial list of inconsistencies that "diskrepair" checks for.

1. Blocks duplicated in files or free list
2. Out-of-range blocks or fdns
3. Missing blocks
4. Bad free list
5. Unreferenced files
6. Inactive fdns
7. Unknown fdn type

(continued)

8. Incorrect link counts
9. Incorrect free block or free fdn count
10. Invalid sizes in System Information Record

Unreferenced files are handled in one of two ways. First, an attempt is made to give the file a name by putting it into the directory "lost+found" in the root directory of the disk being tested. The name given to the file is of the form "file<fdn>", where <fdn> represents the fdn number of the file. In order for this procedure to work, the directory "lost+found" must already exist on the disk being checked, and it must have room for the entry. (This directory is created when the user makes a system disk.) Initially, the lost-and-found directory has room for thirty-two entries--including "." and "..". A user who wants to increase the capacity of the lost-and-found directory must first create enough files in it to add another block to its size, then delete the files. This procedure should not be attempted on a disk that is already damaged. If it is not possible to put the unreferenced file into the "lost+found" directory (because there is either no directory "lost+found" or no room in it), "diskrepair" deletes the file (or prompts for permission to delete it if the 'p' option was specified).

If an error is associated with an fdn, a display of pertinent data from that fdn is printed. The display includes the fdn number of the file, its size in bytes, its owner, the time of its last modification, and one of the following types:

b = block device
c = character device
d = directory
f = file
i = inactive
u = unknown

The "diskrepair" utility should generally be run only in single-user mode on an otherwise inactive system. It should never be run on an active disk. If the 'n' option is not specified (the disk may be written to), "diskrepair" attempts to unmount the disk being tested. If the device being tested is the system disk (or root device), the system must be in single-user mode for the utility to perform repairs. If the disk being tested is the system disk, and if a repair is made which requires writing to the System Information Record (block number 1), "diskrepair" stops the system upon completion and issues an appropriate message instructing the user to reboot the operating system. This procedure is necessary to prevent conflicts between the written data and similar data kept in memory.

Detailed descriptions of the options follow.

(continued)

The ´a´ Option

The ´a´ option automatically places any bad blocks found by "diskrepair" in the file "/.badblocks". It also runs "diskrepair" continuously until either the disk is fixed or the program has executed ten times.

The ´b´ Option

The ´b´ option instructs "diskrepair" to run only the "blockcheck" portion of the utility. This procedure is often considerably faster, but still provides a fairly complete assessment of the validity of the disk structure.

The ´f´ Option

The ´f´ option instructs "diskrepair" to run only the "fdncheck" portion of the utility. This option is useful if a problem is suspected in the directory structure, but the result is by no means a thorough check of the structure of the disk.

The ´i´ Option

By default, "diskrepair" does not allow the user to repair the root device (system disk) unless the system is in single-user mode. However, certain systems--such as the Tektronix 4404--cannot operate in single-user mode. In order to repair the system disk on such a system the user must specify the ´i´ option, which tells "diskrepair" to ignore this restriction. Under no circumstances should this option be used on a system which can operate in single-user mode.

The ´m´ Option

The operating system maintains a list of blocks available for use called the free list. A missing block is any block in the volume space which is not a part of any file and is not in the free list. The existence of such blocks is a harmless error in the structure of the disk. "Diskrepair" generally places these blocks in the free list. The ´m´ option, however, instructs "diskrepair" not to rebuild the free list solely on account of missing blocks. This option reduces the time required for "diskrepair" to run if missing blocks are the only problem in the free list.

The ´n´ Option

The ´n´ option tells "diskrepair" to report all errors but to make no attempt to fix them. Therefore, "diskrepair" opens the device for

(continued)

reading only. This option is useful for checking the structure of a disk without risking the loss of data during repairs.

The ´p´ Option

If the user specifies the ´p´ option, "diskrepair" reports each error, followed by a prompt requesting permission for the proposed repair. All prompts require an answer of either ´y´ ("yes") or ´n´ ("no").

Many repairs result in the loss of data. (The user can generally infer what has been lost from the messages "diskrepair" displays.) Judicious use of the ´n´ and ´p´ options not only allows the user to assess the damage to the disk and decide which information may be sacrificed during the repair process but also provides the opportunity to try to salvage the data before repairing the disk.

The ´q´ Option

This option inhibits certain warnings and messages from "diskrepair". Several conditions exist which, while not technically errors in disk structure, may cause problems. These conditions usually result in a warning message; the ´q´ option inhibits them.

The ´r´ Option

By default, if "diskrepair" finds that the free list is in error, it rebuilds it. The ´r´ option instructs "diskrepair" to rebuild the free list whether or not it contains errors. This option is useful if the free list is known to be bad or if the user wants to reduce fragmentation within the list.

The ´u´ Option

The ´u´ option generates a report on the block usage of the specified device. This report is printed at the end of the "diskrepair" operation. It contains statistics on (1) the number of each type of file in the file system and the total number of files in the system, (2) the number of unused blocks and the number of used blocks, including a breakdown of how the used blocks are allocated, and (3) the number of free fdns and the number of fdns in use.

The ´v´ Option

"Diskrepair" operates in one of two modes: simple or verbose. Simple mode is selected by default; verbose mode is selected by the ´v´ option. In simple mode "diskrepair" reports only those errors which require the

(continued)

deletion of either files or directory entries. In verbose mode all errors are reported. In addition, informative messages are printed describing what phase "diskrepair" is performing.

In verbose mode the ´p´ option causes "diskrepair" to prompt for permission regarding all errors. In simple mode the user is prompted only for those errors which require the deletion of either files or directory entries; all other errors are automatically repaired without prompting.

The ´z´ Option

Normally, "diskrepair" reports a possible error in the size of a file. The ´z´ option suppresses such messages. If the ´q´ option is in effect, the ´z´ option is redundant.

EXAMPLES

1. /etc/diskrepair /dev/w0
2. /etc/diskrepair /dev/w0 +n
3. /etc/diskrepair /dev/fd0 +pv
4. /etc/diskrepair /dev/fd0 +ru
5. /etc/diskrepair /dev/fd0 +mq

The first example checks the logical structure of the disk in drive w0. By default, "diskrepair" tries to fix every error it encounters. These repairs may result in the loss of data from the disk.

The second example checks the logical structure of the disk in drive w0, reports those errors which require the deletion of either files or directory entries, but performs no repairs.

The third example checks the logical structure of the disk in floppy drive 0. "Diskrepair" reports all errors it finds and prompts for permission before making any repairs.

The fourth example checks the logical structure of the disk in floppy drive 0. "Diskrepair" rebuilds the free list no matter what and prints a summary of block usage when it is finished.

The fifth example also checks the logical structure of the disk in floppy drive 0. It does not rebuild the free list solely on account of missing blocks. Neither does it print the warnings and messages that result from problems which are not technically errors in the structure of the disk but which may cause problems.

(continued)

NOTES

. "Diskrepair" cannot solve all the problems a disk may have. For example, it does not methodically search for and repair physical media problems (but see the ´a´ option). As for problems with the logical structure of the disk, "diskrepair" can only repair an error if the damaged information is redundant--that is, if there is some way of determining what the information should be. It cannot, for instance, fix a badly damaged SIR; nor can it repair a disk if the root directory is severely damaged. It is therefore imperative that up-to-date backups of all important files be maintained.

ERROR MESSAGES

Blockcheck terminated abnormally.
"Blockcheck" received a program interrupt from the operating system. The user cannot determine the source of such an error; however, it is not indicative of a problem with either "diskrepair" or the device. "Diskrepair" should be rerun, for the problem may not recur.

Can´t call /etc/blockcheck.
"Diskrepair" cannot read or execute the file "/etc/blockcheck".

Can´t call /etc/fdncheck.
"Diskrepair" cannot read or execute the file "/etc/fdncheck".

Can´t determine mode.
By default, "diskrepair" cannot repair the root device unless the system is in single-user mode (see the ´i´ option). Therefore, if the device specified on the command line is the root device, "diskrepair" must determine what mode the system is operating in. This error message means that "diskrepair" cannot access the file "/act/utmp" (it is the length of this file that tells what mode is in effect).

Can´t read System Information Record.
The SIR is so badly damaged physically that "diskrepair" cannot read it. The user may be able to salvage some information from the disk, but must eventually reformat it.

Can´t stat root.
"Diskrepair" cannot read the fdn which describes the root directory. The user may be able to salvage some information from the disk, but must eventually reformat it.

Can´t stat std. output.
"Diskrepair" cannot read the fdn of whatever file is open as standard output. The user should rerun "diskrepair" with the terminal as standard output.

(continued)

Conflicting options.
The options specified on the command line conflict with each other.

Device is busy.
Any alterations that "diskrepair" makes must be made when the disk is not in use. Therefore, "diskrepair" determines whether or not the specified disk is mounted, and, unless the user specifies the 'n' option, it tries to unmount a mounted disk before proceeding. This error message means that either some user's working directory is on the specified disk or some user is accessing a file on that disk.

Disk needs repair!
The structure of the disk is not logically sound. The user should rerun "diskrepair" to correct the problems.

Error reading block <block_num>.
Error reading fdn <fdn_number> in block <block_num>.
Error writing block <block_num>.
Error writing fdn <fdn_num> in block <block_num>.
"Diskrepair" encountered a physical error on the disk. If the 'a' option is in effect, "diskrepair" calls the program "/etc/badblocks", which places the offending block in the file "/.badblocks". If the 'p' option is also in effect, "diskrepair" prompts the user for permission to call this utility. The user should grant permission.

If the 'a' option is not in effect, but either the 'p' or 'n' option is in effect, "diskrepair" prompts for permission to continue. If the user chooses to continue when the 'n' option is not in effect, the results are entirely unpredictable. They depend on precisely which block is damaged. Continuing with "diskrepair" may cause further damage to the disk. We suggest that the user respond negatively to the prompt to continue whenever "diskrepair" reports an I/O error and immediately rerun "diskrepair" with the 'a' option.

ERROR UPDATING SIR. DISK IS BAD!
"Diskrepair" encountered an I/O error when it tried to make the necessary changes in the SIR. The user should try again to execute "diskrepair". If the error persists, the user cannot salvage any of the data on the disk.

/etc/blockcheck is invalid.
The version of the "blockcheck" command is not the correct one.

/etc/fdncheck is invalid.
The version of the "fdncheck" command is not the correct one.

(continued)

Fdncheck terminated abnormally.
"Fdncheck" received a program interrupt from the operating system. The user cannot determine the source of such an error; however, it is not indicative of a problem with either "diskrepair" or the device. "Diskrepair" should be rerun, for the problem may not recur.

Intentional system stop. Reboot UniFLEX.
If the SIR of the root device must be updated, "diskrepair" kills all tasks running on the system and locks up the system so that no new tasks can begin. It then modifies the SIR. This procedure is necessary to prevent conflicts between the written data and similar data kept in memory. After updating the SIR, "diskrepair" stops the system and prints this error message. The user must reboot the system before proceeding.

Must be in single-user mode.
By default, "diskrepair" cannot repair the root device unless the system is in single-user mode (see the 'i' option).

No device specified.
The user did not specify a device on the command line.

No such device.
The user specified a nonexistent device on the command line.

Not a block device.
"Diskrepair" can only operate on block devices.

Output directed to device under test.
When testing the structure of a disk, it is impractical to try to redirect the output (the results of the test) to a file on the disk being tested. The user should reexecute "diskrepair" without redirecting the output or redirecting it to a different, mounted device.

Permission denied.
A user who executes "diskrepair" without the 'n' option must have both read and write permission on the specified device. A user who executes "diskrepair" with the 'n' option needs only read permission.

Problems encountered. Diskrepair should be rerun.
"Diskrepair" may encounter more problems than it can fix during one run. For example, it can only handle a certain number of duplicate or out-of-range blocks. If "diskrepair" cannot fix all the errors it encounters, or if it encounters an I/O error but continues operation, it prints this error message when it finishes.

(continued)

Unknown option: '<char>'
The option specified by <char> is not a valid option to the "diskrepair" command.

Unmount error: <error_num>
"Diskrepair" encountered some problem other than a busy device when it tried to unmount the device. The accompanying error number is the number of the UniFLEX error that caused the failure. The user should consult the operating system manual for an explanation of the error.

SEE ALSO

badblocks
blockcheck
fdncheck

diskrepair

Check and, optionally, repair inconsistencies in the structure of a disk.

SYNTAX

/etc/diskrepair <device_name> ... [+<bfmnpqruv>]

Arguments

<device_name> Name of the device to check.

Options

b	Perform "blockcheck" only.
f	Perform "fdncheck" only.
m	Ignore missing blocks.
n	Do not attempt to fix errors.
p	Prompt for permission to repair.
q	Use quiet mode.
r	Rebuild the free list whether or not it is in error.
u	Report on the usage of disk blocks.
v	Use verbose mode.

DESCRIPTION

The "diskrepair" utility checks the structure of the disk or disks specified by <device_name>. The structure of the disk refers to the layout of and the connections among files, directories, free space, swap space, and other information that makes up the file system. Any inconsistencies in the structure are reported and, optionally, repaired. "Diskrepair" does not check or repair media errors (I/O errors).

While it is operating, "diskrepair" calls two other utilities--"blockcheck" and "fdncheck", which are both located in the directory "/etc". "Blockcheck" is concerned with the allocation of blocks on the disk. It locates problems such as duplicate blocks, missing blocks, and invalid block addresses. "Fdncheck" is concerned with the directories on the disk. It locates problems such as unreferenced files, file names with invalid associated files, and so forth.

There are two major modes of operation, simple and verbose. The simple mode is selected by default; the verbose mode is selected by the "+v" option. In the verbose mode "diskrepair" reports all detected errors. In the simple mode it reports only those errors which require the deletion of files or of directory entries. If executed in simple mode, "diskrepair" issues a message upon completion which informs the user whether or not the disk is in need of repair.

By default, all detected errors are automatically repaired (if possible). Two options exist to alter the handling of errors. The "+n" option instructs "diskrepair" not to repair any errors. The "+p" option instructs "diskrepair" to prompt the user for permission to repair the errors it reports. In verbose mode this option causes "diskrepair" to prompt the user regarding all errors. In the simple mode, the user is prompted only for those errors which require the deletion of files or of directory entries; all other errors are automatically repaired without prompting. It should be noted that most repairs result in a loss of data. The user can generally infer which data have been lost from the messages displayed.

When using the command in simple mode (without the "+v" option), the user need not understand what types of checks are made by "diskrepair". The only decisions required are whether or not to delete the reported files. In verbose mode, much more information is given to the user. While this document is not intended to give full details of this information, the following list shows most of the inconsistencies in disk structure for which "diskrepair" checks. First, however, a few definitions are in order. A "file descriptor node" (or "fdn") is an area on the disk which contains all the information the system needs about a file. There should always be one fdn per file on the disk. A UniFLEX directory entry is simply a file name and a pointer to the proper fdn. There may be multiple directory entries pointing to the same fdn (multiple names for the same file). Each pointer to an fdn is called a "link" to that file. If there is a file with no links, it is considered to be "unreferenced". "Out-of-range" refers to a pointer to a disk block or to an fdn which is beyond the valid number of blocks or fdns for the disk being tested. Here now, is a partial list of inconsistencies that "diskrepair" checks for.

1. Blocks duplicated in files or free list
2. Out-of-range blocks or fdns
3. Missing blocks
4. Bad free list
5. Unreferenced files
6. Inactive fdns
7. Unknown fdn type
8. Incorrect link counts
9. Incorrect free block or free fdn count
10. Invalid sizes in System Information Record

Unreferenced files are handled in one of two ways. First, an attempt is made to give the file a name by putting it into the directory "lost+found" in the root directory of the disk being tested. The name given to the file is of the form "file<fdn>", where <fdn> represents the fdn number of the file. In order for this procedure to work, the directory "lost+found" must already exist on the disk being checked, and it must have room for the entry. Therefore, prior to running "diskrepair", the user must create "lost+found" in the root directory of the disk being tested. He must also create empty slots for entries

by creating a number of files and then deleting them. If it is not possible to put the unreferenced file into the "lost+found" directory (because there is either no directory "lost+found" or no room in it), "diskrepair" deletes the file (or prompts for permission to delete it if "+p" was specified).

If an error is associated with an fdn, a display of pertinent data from that fdn is printed. The display includes the fdn number of the file, its size in bytes, its owner, time of its last modification, and one of the following types:

b = block device
c = character device
d = directory
f = file
i = inactive
u = unknown

The "diskrepair" utility should generally be run only in single-user mode on an otherwise inactive system. It should never be run on an active disk. If the "+n" option is not specified (the disk may be written to), "diskrepair" attempts to unmount the disk being tested. If the device being tested is the system disk (or root device), the system must be in single-user mode for the utility to perform repairs. If the disk being tested is the system disk, and if a repair is made which requires writing to the System Information Record (block number 1), "diskrepair" stops the system upon completion and issues an appropriate message instructing the user to reboot the operating system. This procedure is necessary to prevent conflicts between the written data and similar data kept in memory.

Complete descriptions of the options follow.

The "+b" Option

The "+b" option instructs "diskrepair" to run only the "blockcheck" portion of the utility. This procedure is often considerably faster, but still provides a fairly complete assessment of the validity of the disk structure.

The "+f" Option

The "+f" option instructs "diskrepair" to run only the "fdncheck" portion of the utility. This option is useful if a problem is suspected in the directory structure, but the result is by no means a thorough check of the disk structure.

The "+m" Option

Ignore missing blocks. UniFLEX maintains a list of sectors available for use called the free list. Missing blocks are those disk blocks which are not a part of any file and are not in the free list. The

existence of such blocks is an error in the disk structure. "Diskrepair" generally places these blocks in the free list. Missing blocks are not harmful to the disk structure, however, and the "+m" option instructs "diskrepair" not to rebuild the free list because of them. This reduces the time required for "diskrepair" to run if missing blocks are the only problem in the free list.

The "+n" Option

Report all errors, but make no attempt to fix them. The disk is opened for reading only.

The "+p" Option

Use prompt mode. If an error is found on the disk, it is reported, followed by a prompt requesting permission for the proposed repair. All prompts require an answer of either 'y' ("yes") or 'n' ("no").

The "+q" Option

Use quiet mode. This option inhibits certain warnings and messages. Several conditions exist which, while not technically errors in disk structure, may cause problems. These conditions usually result in a warning message; the "+q" option inhibits them.

The "+r" Option

Rebuild the free list. If "diskrepair" finds that the free list is in error, it may rebuild it, depending on the options specified. The "+r" option instructs "diskrepair" to rebuild the free list whether or not it contains errors. This is useful if the free list is known to be bad or if the user wants to reduce fragmentation within the list.

The "+u" Option

Report usage of disk blocks. The "+u" option generates a report on the block usage of the specified device. This report is printed at the end of the "diskrepair" operation.

The "+v" Option

Use verbose mode. In verbose mode all errors are always reported. In addition, informative messages are printed describing what phase "diskrepair" is performing. Without this option, only those errors requiring the deletion of files or of directory entries are reported.

SEE ALSO

blockcheck
fdncheck
System Maintenance Guide

dperm

Assign the value to be used for default permissions assignment.

SYNTAX

```
dperm [u-rwx] [o-rwx]
```

DESCRIPTION

The dperm command is used to set the default permissions. Everytime a file is created on the system, it is assigned a set of permission bits which will determine whether or not the user and others may read, write, or execute the file. The permissions assigned depend on the program which is creating the file. The editor, for example, creates all files with 'rw-rw-' permissions, which allows the owner as well as others to read and write the file, but not execute it. The dperm command allows one to instruct the system to always deny certain permissions, independent of how the file is created. It is possible to have any of the 'rwx' bits turned off for the owner of the file, as well as for others. A few examples will demonstrate how this is accomplished.

```
dperm o-rwx
dperm u-w o-wx
dperm
```

The first example sets the default such that whenever a file is created, all permissions for others will be denied. The second example will always disable write permission for the user (owner) and will also disable write and execute permission for others. The last example will set the default to null, which means there are no default permissions (this is the normal state). The defaults do not apply to permissions set by the 'perms' command. The dperm command is part of the shell and does not reside on the disk.

SEE ALSO

perms

echo

Echo (display) the command arguments on the standard output.

SYNTAX

```
echo [+l] [+hex] [argument]
```

DESCRIPTION

Echo is useful for displaying diagnostic messages to the terminal during long shell scripts as well as setting up crt terminal control character strings for terminal configuration. The command simply displays each argument it encounters on the standard output. If the argument starts with a plus sign, it is not echoed. A '+l' argument causes the carriage return normally output at the end of the echo command to be suppressed. If the value following the '+' is a hex number (maximum of hex 7f), The equivalent byte will be output. This is useful for producing special control character sequences. Several examples will demonstrate the use of echo.

```
echo This is a test of echo.
echo +l Hello
echo +l +7 +d +a
```

The first example will print the string 'This is a test of echo.' on the terminal. The second line will print 'Hello' without a carriage return following it. The last example will output a bell character (7), a carriage return (d), and a line feed (a). Only one carriage return will be output since echo will suppress the one it normally outputs (because of the +l option).

SEE ALSO

shell

edit

Invoke the text editor to modify an existing text file or create a new text file.

SYNTAX

```
edit [+bny]
edit file [+bny]
edit file1 file2 [+bny]
```

DESCRIPTION

The first form of the edit command is used to create a new file whose name will be requested by the editor at the end of the editing session. The editor will prompt for the file name before returning control to the operating system.

The second form of the edit command requests the creation of a new file named 'file' or the editing of an existing file named 'file'. If the file specified as the argument does not already exist, it is assumed that the name is that of a new file which is to be created. If the argument is the name of an existing file, it is assumed that the file is to be edited.

The third form of the edit command is used to edit an existing file named 'file1', with the revised version being written into the file named 'file2'.

The options are allowable on all forms of the call. All option letters must be specified in lower case. Option letters other than those indicated are ignored with a warning message being issued. The options, and their implications, are:

+b Do not create a backup file.
When editing an existing file, the original copy will not be saved in a backup file at the end of the editing session.

+y Delete any existing backup file.
If a backup file already exists at the end of the editing session, it is automatically replaced by a new backup file containing the original copy of the file being edited. If the editor was called with two arguments, as in the third form above, and the second file already exists, this option will cause the deletion of that file at the beginning of the editing session.

+n Do not read in the text.
At the beginning of the editing session, the information at the

beginning of the file being edited is not read into the edit buffer, leaving it available for large insertions which are to appear at the beginning of the file.

SEE ALSO

Text Editor Manual

fdncheck

This command is used to check the integrity of the fdn structure on a disk.

SYNTAX

```
/etc/fdncheck block_device_name
```

DESCRIPTION

The 'block_device_name' is the name of a block device. Two examples will demonstrate its use.

```
/etc/fdncheck /dev/fd0
/etc/fdncheck /dev/hd0
```

The first example will check the fdn structure on floppy disk drive 0, and the second example will check the hard disk. Any output generated by this command indicates a bad disk. The meanings of the error messages will not be described here. Any disk which produces output from this command should no longer be used! There are no exceptions.

SEE ALSO

blockcheck, devcheck

flex

Copy a file from a FLEX format disk into a UniFLEX file.

SYNTAX

```
flex
flex FLEXFILE uniflexfile [+r]
```

DESCRIPTION

This command allows text and data files from FLEX format disks to be copied to UniFLEX disks. It may only be run from the single-user mode of UniFLEX. The FLEX disk must always be placed in floppy disk drive #1 (the second drive unit). This command will only work with single density FLEX disks. Double sided ones will work correctly. There are two forms of 'flex'. The first, where no parameters are supplied, lists the name and size in sectors of all files on the FLEX disk. The second form permits a single file to be copied from the FLEX disk into a UniFLEX file. The FLEX filename specification should conform to the standard FLEX filename rules and defaults to a '.TXT' extension. If the '+r' option (raw mode) is specified, the file will be copied as is, byte for byte. Without the '+r' option the file will be assumed to be textual and FLEX space expansion will take place. One exception is that FLEX random type files will automatically be copied in the raw mode even though no '+r' option is given. Note also that the sector map sectors associated with FLEX random files will be skipped (not copied). Two examples of the second form follow:

```
flex MANUAL manual
flex ADDRESS.DAT addrdata +r
```

The first copies the file 'MANUAL.TXT' from the FLEX disk into a UniFLEX file called 'manual'. The second copies the FLEX file 'ADDRESS.DAT' into a UniFLEX file called 'addrdata' using the raw mode such that the file is copied without space expansion.

format

Format a disk for system use.

SYNTAX

/etc/format [+BdflLmMnPqrv]

Options

B	Write boot sector only. Do not format disk.
d=<file_name>	Format the device <file_name>.
f=<blocks>	Establish <blocks> fdn blocks.
l=<file_name>	Take bad sector addresses from <file_name>.
L	Take bad sector addresses from terminal.
m=<model_code>	Use <model_code> for disk parameters.
M	Take disk parameters from terminal.
n	Do not prompt for information on disk volume.
P	Prompt for disk parameters.
q	Use quiet mode.
r=<swap>	Reserve <swap> cylinders for swap space.
v	Verify disk after formatting.

DESCRIPTION

The family of "format" commands is used to prepare disks for system use. This document is a general description of all "format" commands; specific "format" commands are documented elsewhere. The specific commands generally have a suffix appended to the general name "/etc/format". They perform media formatting as well as UniFLEX structural formatting. Only the system manager may execute this command.

If the system is unable to collect the physically contiguous memory required by the hardware to format a disk, an appropriate message is issued and the user is asked if he wishes to wait. If he does, the program sleeps for a short time and then tries again. If the required memory is repeatedly unavailable, it may be necessary to go into single-user mode where the reduced number of tasks almost always yields enough memory for formatting.

Complete descriptions of the options follow.

The "+B" Option

A "+B" option on the "format" command line instructs the "format" command to write the boot sector onto the specified disk without first formatting the disk. The system assumes that the specified disk has

previously been formatted. It writes only the boot sector (sector number 0).

The "+d" Option

Each specific "format" command operates on a particular disk unit unless another device is explicitly specified. If this default disk has been mounted (using the "mount" command), it is automatically unmounted prior to the formatting operation. The "+d" option allows the user to specify the device to be formatted. Its syntax is

d=<file_name>

where <file_name> is the complete file specification of the desired block device.

The "+f" Option

Formatted UniFLEX disks use fdn blocks to hold information about files on the disk. Each fdn block contains 8 fdns. By default, "format" uses 3% of the total disk space for fdn blocks. This default value may be overridden by the "+f" option, which allows the user to specify the decimal number of fdn blocks to establish on the disk. At least one block must be allocated for fdns on every formatted disk.

The "+l" and "+L" Options

If "format" detects media errors (see "+v" option), it notifies the user of these errors and automatically omits the faulty sectors from the available space on the disk by placing them in a file called ".badblocks" in the root directory of the disk. As a result, any program which physically accesses blocks from the disk (by directly reading blocks from the device rather than by opening and accessing them from a file) receives an I/O error if it accesses any of these bad blocks. Newer versions of programs like "devcheck", which do access the disk physically, are aware of the existence of the ".badblocks" file and ignore all the blocks contained in that file.

It is also possible to specify in advance a list of bad sectors which "format" should omit from the available space. "Format" places the specified sectors in the file ".badblocks", thus omitting them from the available space on the disk. This list is specified via the "+l" or "+L" option. These options are most commonly used on hard disks. Hard disk manufacturers often provide a list of bad spots that have been detected on a particular drive. The "+l" and "+L" options allow the user to be certain that such spots are considered bad by "format". The bad spots must be specified as logical, 512-byte sector addresses. If the manufacturer's list of bad spots is given in 40-byte ranges or in 256-byte sectors, the user should refer to Appendix B for conversion to physical sector numbers. The physical sector numbers must then be converted to logical sector numbers, which take into account the sector interleaving. Appendix C contains a table of conversions from physical

to logical sector numbers.

Using the "+l" or "+L" option, the user then specifies the bad sectors to "format" by means of a head address, a cylinder address, and a logical sector number. The syntax for identifying a bad sector is as follows:

<HD>/<CYL>/<SC>

where <HD> is a two-digit, decimal number representing the head number or surface number (0 through (no. of heads - 1)); <CYL> is a three-digit, decimal number representing the cylinder number (0 through (no. of cylinders - 1)); and <SC> is a two-digit, decimal number representing the logical, 512-byte sector number (1 through 17 for mini-Winchesters). For example, head number 1, cylinder number 57, sector number 2, is specified as

01/057/02

These bad sector addresses may be specified from either the terminal or from an existing disk file. If they are to be specified from the terminal, the "+L" (uppercase) option should be used. "Format" then prompts the user for bad sector addresses, one per line, before beginning the formatting procedure. The list is terminated by typing a control-D as the first character of a line. If the bad sectors are to be specified from a disk file, the "+l" (lowercase) option should be used. The syntax for this option is

+l=<file_name>

For example, if the file containing the bad sector addresses is called "bad-spots" and is located in the "/etc" directory, the option to use is:

+l=/etc/bad-spots

This file should contain one sector address per line. Each address should conform to the format described previously.

The "+m" and "+M" Options

Specific parameters about disk type and size are supplied through the "+m" or "+M" (model) option. The syntax for the "+m" option is

m=<model_code>

where <model_code> is coded information about the particular drive type and size. A list of valid codes is provided with the documentation of each "format" command. The "+M" option does not take parameters on the commmand line, but instead causes "format" to prompt the user for the information. The user's response should be in the same format as that described for the "+m" option. In addition, the "+M" option asks to display a list of the models it knows about and their associated codes. The user can then use this information with the current "+M" option or

with "+m" options on subsequent "format" commands. If neither the "+m" nor the "+M" option is specified, the model defaults to the standard drive supplied with the hardware. This default drive's coded information is displayed before the formatting actually begins.

The "+n" Option

The "format" command prompts the user for a 'file system name', 'volume name', and 'volume number' unless the "+n" option is specified to inhibit such prompts.

The "+P" Option

The "+P" option may be specified in place of the "+m" or "+M" option. It tells "format" to prompt the user for the disk parameters. These parameters include the number of cylinders, the number of heads, the cylinder number at which to begin reduced write, the cylinder number at which to begin write precompensation, and the stepping (seek) rate. All responses should be in decimal unless otherwise indicated. This option is automatically invoked if a "+m" option is specified with an unknown code.

The "+q" Option

Before actually starting to format the disk, "format" normally sends a prompt to the terminal to ask if the user is ready to continue. The "+q" (quiet) option supresses this prompt. In addition, "+q" inhibits all informative messages from "format" if no errors are encountered during formatting.

The "+r" Option

The "+r" option is used to reserve swap space on the disk. The syntax for this option is

r=<swap>

where <swap> is a decimal number which specifies the number of cylinders to reserve. For example, "+r=20" reserves 20 cylinders for swap space. (For a definition of "cylinder" and other terms, see Appendix A.) Although the amount of swap space needed depends heavily on the specifics of the particular installation, the minimum recommended swap space is approximately 400 Kbytes. Appendix D, which lists the cylinder sizes of various disks, can be used to determine the number of cylinders to reserve. Note that swap space is not necessary on data disks and that by default, "format" does not reserve swap space.

The "+v" Option

The "+v" (verify) option instructs "format" to verify the media after formatting. If this option is specified, "format" individually verifies every sector on the disk by writing an arbitrary pattern to the sector and then attempting to read it correctly. It reports any sectors which fail this test to the user. On large disks, a considerable length of time is required to perform this verification. The need for the "+v" option depends on the hardware involved and is discussed in the documentation of each specific "format" command. The option is often desirable when the user is formatting a floppy disk because floppies do not automatically verify all written data. Some Winchester disk units perform automatic verification of all written data; others do not.

Appendix A
Definitions of Disk Drive Terminology

The following definitions apply to this document.

Head: A device which transfers data from one disk surface. Normally there is one head per recording surface.

Track: All the sectors accessible by one head on one surface without moving the head assembly.

Cylinder: All tracks accessible by all heads without moving the head assembly. Cylinder address is the corresponding position of the head assembly.

Standard 8-inch floppy disks have 77 cylinders, numbered 0 through 76. The distinction between a track and a cylinder on a floppy is sometimes overlooked. The term track is often incorrectly used in place of cylinder.

Appendix B
Conversion to Physical Sector Numbers for Mini-Winchesters

The following tables can be used to convert a 40-byte range or 256-byte physical sector number into the 512-byte physical sector number required by "format" for mini-Winchesters. A single 40-byte range or 256-byte sector may cross a 512-byte sector boundary and therefore map onto two 512-byte sectors.

Table B-1. Conversion from 40-byte Range to 512-byte Sectors

40-byte Zone	512-byte Sector(s)
0-10	1
11-20	1-2
21-30	2-3
31-40	3
41-50	3-4
51-60	4-5
61-70	5
71-80	5-6
81-90	6-7
91-100	7
101-110	7-8
111-120	8-9
121-130	9
131-140	9-10
141-150	10-11
151-160	11
161-170	11-12
171-180	12-13
181-190	13-14
191-200	14
201-210	14-15
211-220	15-16
221-230	16
231-240	16-17
241-250	17

Table B-2. Conversion from 256-byte Sectors to 512-byte Sectors

256-byte Sector	512-byte Sector(s)
1	1
2	1-2
3	2
4	2-3
5	3
6	3-4
7	4
8	4-5
9	5
10	5-6
11	6
12	6-7
13	7
14	7-8
15	8-9
16	9
17	9-10
18	10
19	10-11
20	11
21	11-12
22	12
23	12-13
24	13
25	13-14
26	14
27	14-15
28	15
29	15-16
30	16-17
31	17
32	17

Appendix C
Conversion from Physical to Logical Sector Number

The following table gives the logical, 512-byte sector number that corresponds to a given physical, 512-byte sector for mini-Winchester disks. To use this table, the user must know the sector interleave used by the specific "format" command.

Table C-1. Conversion from Physical to Logical Sector Number

Physical Sector No.	Logical Sector No. for Interleave of								
	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
2	10	7	14	8	4	6	16	3	13
3	2	13	10	15	7	11	14	5	8
4	11	2	6	5	10	16	12	7	3
5	3	8	2	12	13	4	10	9	15
6	12	14	15	2	16	9	8	11	10
7	4	3	11	9	2	14	6	13	5
8	13	9	7	16	5	2	4	15	17
9	5	15	3	6	8	7	2	17	12
10	14	4	16	13	11	12	17	2	7
11	6	10	12	3	14	17	15	4	2
12	15	16	8	10	17	5	13	6	14
13	7	5	4	17	3	10	11	8	9
14	16	11	17	7	6	15	9	10	4
15	8	17	13	14	9	3	7	12	16
16	17	6	9	4	12	8	5	14	11
17	9	12	5	11	15	13	3	16	6

Appendix D
Cylinder Sizes for Various Disks

The "+r" option requires the specification of the number of cylinders to reserve for swap space. To calculate this number the user must know how much swap space he wants and the size of each cylinder.

Mini-Winchester disks contain approximately 8.5 Kbytes of storage per track. The number of tracks per cylinder is dependent on drive type and can be obtained from the hardware documentation or from the documentation of the specific "format" command. Note that the number of tracks per cylinder is the same as the number of heads in the drive. For example, consider a Computer Memories model 5619 mini-Winchester disk. This device has 306 cylinders and six heads. Twenty cylinders are required to reserve 1 Mbyte of swap space (20 cyl. * 6 tracks/cyl. * 8.5 Kbytes/track).

The following table provides the necessary cylinder sizes for floppy disks.

Table D-1. Cylinder Sizes for Various Formats of Floppy Disks

		8" Disk		5" Disk	
Sides	Density	Sectors per cyl	Kbytes per cyl	Sectors per cyl	Kbytes per cyl
Single	Single	8	4	5	2.5
Single	Double	16	8	10	5
Double	Single	16	8	10	5
Double	Double	32	16	20	10

formatfd

Format an 8-inch floppy disk for use on a SWTPc system.

SYNTAX

/etc/formatfd [+BdflLmMnqrv]

Options

B	Write boot sector only. Do not format disk.
d=<file_name>	Format the device <file_name>.
f=<blocks>	Establish <blocks> fdn blocks.
l=<file_name>	Take bad sector addresses from <file_name>.
L	Take bad sector addresses from terminal.
m=<model_code>	Use <model_code> for disk parameters.
M	Take disk parameters from terminal.
n	Do not prompt for information on disk volume.
q	Use quiet mode.
r=<swap>	Reserve <swap> cylinders for swap space.
v	Verify disk after formatting.

DESCRIPTION

The "formatfd" command is used to prepare an 8-inch floppy disk for system use. The options are more fully explained in the documentation for the "format" command. Certain aspects of the options, which specifically apply to "formatfd", are presented here.

If the "+d" option is not used, the default disk unit is "/dev/fd1".

When using the "+l" or "+L" option, it is not necessary to convert physical sector numbers to logical sector numbers because no sector interleave is used.

Table 1, which follows, refers to "side bits". There is a bit maintained in each track of a double-sided floppy disk which indicates the side of the disk (side 0 or side 1). In SWTPc hardware these bits are optional--they can either be enabled or disabled. Thus, it makes no difference whether you format a disk with side bits or without. Some hardware from other vendors, however, requires side bits, and to be compatible with this hardware the user should enable side bits. It should be noted that some versions of SWTPc UniFLEX prior to version X.09 cannot support side bits. Therefore, to be compatible with certain old versions of UniFLEX, the user should disable side bits.

If neither the "+m" nor "+M" option is used, the default model code is FD-SS. The following table shows the model codes that may be used with these options.

Table 1. Model Codes for 8-inch Floppy Disks

Model Code	Sides	Density	Side Bits
FD-SS	Single	Single	
FD-SD	Single	Double	
FD-DS	Double	Single	Disabled
FD-DD	Double	Double	Disabled
FD-DS+	Double	Single	Enabled
FD-DD+	Double	Double	Enabled

Note that "formatfd" does not support the "+P" option because the disk parameters in question do not vary from floppy disk to floppy disk.

The "+v" option instructs "formatfd" to verify the floppy disk after formatting. Floppies do not automatically verify all written data, so in order to verify a disk, this option must be used.

SEE ALSO

format

formatwd1000

Format a mini-Winchester disk for use on a SWTPc system.

SYNTAX

```
/etc/formatwd1000 [+BdflLmMnPqrv]
```

Options

B	Write boot sector only. Do not format disk.
d=<file_name>	Format the device <file_name>.
f=<blocks>	Establish <blocks> fdn blocks.
l=<file_name>	Take bad sector addresses from <file_name>.
L	Take bad sector names from terminal.
m=<model_code>	Use <model_code> for disk parameters.
M	Take disk parameters from terminal.
n	Do not prompt for information on disk volume.
P	Prompt for disk parameters.
q	Use quiet mode.
r=<swap>	Reserve <swap> cylinders for swap space.
v	Verify disk after formatting.

DESCRIPTION

The "formatwd1000" command is used to prepare a mini-Winchester disk for system use. The options are more fully explained in the documentation for the "format" command. Certain aspects of the options, which specifically apply to "formatwd1000", are presented here.

If the "+d" option is not used, the default disk unit is "/dev/w0".

When using the "+l" or "+L" option, physical sector numbers must be converted to logical sector numbers. In the present version of the core of "formatwd1000" the sector interleave is 5. In all previous releases the sector interleave was 2. Consult Table C-1 in the documentation for "format" to make the conversion.

If neither the "+m" nor "+M" option is used, the default model code is CMI-5619. The following table shows the manufacturer, model code, number of cylinders, and number of heads for the drives supported by this utility. This table may be expanded from time to time. For the most up-to-date list of drives supported by "formatwd1000", execute the command with a "+M" option and request a display of possible model codes.

(cont.)

Table 1. Drives Supported by "formatwd1000"

Manufacturer	Model Code	Cylinders	Heads
Ampex	PYX-13	321	4
Ampex	PYX-20	321	6
Ampex	PYX-27	321	8
Atasi	AT-3020	645	3
Atasi	AT-3033	645	5
Atasi	AT-3046	645	7
Computer Memories Inc.	CMI-5619	306	6
Computer Memories Inc.	CMI-5640	640	6
Rodime	RO-201	321	2
Rodime	RO-202	321	4
Rodime	RO-203	321	6
Rodime	RO-204	321	8
Rotating Memory Systems	RMS-506	153	4
Rotating Memory Systems	RMS-509	215	4
Rotating Memory Systems	RMS-512	153	8
Rotating Memory Systems	RMS-518	215	8
Seagate Technology	ST-506	153	4
Seagate Technology	ST-412	306	4

The "+v" option instructs "formatwd1000" to verify the hard disk after formatting. At the time of this writing (August, 1983) none of the SWTPc mini-Winchester disk systems automatically verifies all written data. Therefore, in order to verify a disk, the "+v" option must be used.

SEE ALSO

format

Format disk media for system use.

SYNTAX

```
/etc/formatfd [+nsdrN]
/etc/formathd
```

DESCRIPTION

The format programs are used to format disk media for system use. The programs perform media formatting as well as structural formatting. All disks are created with a root directory. Only the system manager may execute these commands and they will only run in single user mode.

The floppy disk formatter (/etc/formatfd) will format a disk in drive 1 only (not drive 0). Any mounted disk in this drive will automatically be unmounted. The default disk type is single sided, single density. Prompts for 'file system name', 'volume name', and 'volume number' will be issued unless the '+n' option is specified. The '+s' option will force double sided formatting and the '+d' option will force double density. There is a default number of fdn blocks assigned to the disk but this value may be specified by a decimal number in the option list (this is represented as +N in the syntax listing above). There are 8 fdns per block. There must be at least one block for fdns allocated on the disk. The '+r' option is used to reserve swap space on the disk. The 'r' should be immediately followed by a decimal number which specifies the number of tracks to be reserved. As an example, '+r16' would reserve 16 tracks for swapping. The number of blocks per track depends on the density and side count. Single sided, single density has 8 blocks per track; Double sided has 16 per track. Going to double density will double these values. Remember that a block contains 512 bytes and that the minimum recommended swap space is 200K. The error "Can't obtain special memory" means the system was unable to collect the required physically contiguous memory needed by the hardware to format. To overcome this problem, format floppies in single user mode only.

The hard disk formatter (/etc/formathd) is currently configured to format the device '/dev/hd0' and no others. No options are available. This routine automatically reserves about one megabyte of swap space for system use. This program requires about 30 minutes to execute.

SEE ALSO

System Manager's Guide

free

Report the amount of free space available on the specified devices.

SYNTAX

```
free device ...
```

DESCRIPTION

Free is used to report the amount of free space remaining on specified devices. The total number of free blocks is reported, as well as the number of 'fdns' remaining. This last number is an indication of the number of 'file slots' left and determines how many more files may be created on the device (assuming there are sufficient free blocks remaining). As an example:

```
free /dev/disk0
```

will report the free space available on the device named '/dev/disk0'.

help

This command is used to obtain help with the various user commands on the system.

SYNTAX

```
help [name] ...
```

DESCRIPTION

The help command allows one to get a brief description of the use and syntax of various commands available on the system. Typing 'help' without any arguments will display the names of all commands for which help is available. After the display, a prompt will appear which requests the name of the command desired. If a carriage return is typed, the help command will terminate. Help may also be called with the name or names of commands you need information about following 'help'. Those commands descriptions will be listed if available. A few examples will demonstrate the use of help.

```
help
help dir perms
```

The first example will display the list of commands and then allow the user to select which ones wanted. The second example will print the information available for the 'dir' and 'perms' commands.

history

The history command will display the details of the recent activity of the system.

SYNTAX

history [file]

DESCRIPTION

If no file is specified, the '/act/history' file is used by default. The system is capable of keeping details about certain activity. Typing 'history' will display a record of these recent actions. Each line displayed will start with two characters which are interpreted as follows:

bt	the system was booted
su	the system entered single user mode
mu	the system entered multi-user mode
st	the system was stopped
bd	before date status
ad	after date status

Following each of the above will be a time and date which specifies when the action took place. The displayed line may also start with a two digit number. This number is a terminal number (terminals are numbered from 00 to 99) and may be followed by a user name and time which indicates that user logged on the system with the specified terminal at that time, or it will be followed by the time only, which indicates a system log-off from that terminal. If a file name is specified to 'history', it should be of the same format as the history file. As an example, the file '/act/attempts' may be viewed to show all illegal login attempts. Note that this command may not be available on every system.

info

Display any information associated with a binary file.

SYNTAX

```
info file
```

DESCRIPTION

The info command is used to display information which may be kept with binary files. This information may include the version number, release date, special notes, or anything else useful about the command. As an example of its use:

```
info /bin/edit
```

will display the version number and release date of the editor (named '/bin/edit').

SEE ALSO

UniFLEX 6809 Assembler Manual

insp

Invoke a printer spooler for the specified device.

SYNTAX

```
/etc/insp spooler_device [+f]
```

DESCRIPTION

The insp command is used to invoke a printer spooler for the device specified. If the '+f' option is specified, the spooler will not print the banner page or issue form feeds for each print job. Only the system manager may execute this command.

SEE ALSO

printer control, System Manager's Guide

install

Install the 'uniflex' file on a disk for booting.

SYNTAX

```
/etc/install file
```

DESCRIPTION

When creating a new system disk, which will be used to boot the system, several things must be done. The first is to copy the version of UniFLEX to the root directory of the new disk. The file should be named 'uniflex' to enable booting. Once the file resides in the root directory, it needs to be 'installed'. The 'install' command does just that. It should be noted that only copies of the originally supplied 'uniflex' file(s) may be copied and installed. A 'copy of a copy' will not function correctly. If your original disk ever becomes damaged, you will need to get it replaced (see the 'System Manager's Guide' for more information). The 'file' name specified in this command should always be 'uniflex' or errors will result. For example:

```
++ /etc/install /usr2/uniflex
```

will install the 'uniflex' file in the root directory of the disk mounted on 'usr2'. Only the system manager may execute this command.

SEE ALSO

System Manager's Guide

int

The int command is used to terminate or send a program interrupt to another task.

SYNTAX

```
int [+interrupt] taskid
```

DESCRIPTION

Int will send an 'interrupt' to the task specified by the 'taskid'. If no interrupt is specified, a 'termination interrupt' will be sent (TERMI). A list of program interrupts and their respective numbers appear in the description of the 'cpint' system call. Task id's are reported by the shell and may also be determined by the 'jobs' command. Specifying a task id of 0 in this command will interrupt all tasks associated with the user's terminal (and owned by the current user). Several examples follow:

```
int 2631
int +5 1492
```

The first line will send the termination interrupt to task number 2631. The second line will send the KILLI interrupt (kill) to task number 1492.

SEE ALSO

jobs

jobs

Display the task id's and starting times of all background tasks originated by the caller on the current terminal.

SYNTAX

jobs

DESCRIPTION

For every background task initiated by the user, the shell reports a task id number associated with that task. This task id is needed if it is necessary to terminate or interrupt the task before its completion. The jobs command will display the id number and starting time for all active background tasks originated by the caller on the current terminal. All task numbers are displayed with a preceeding 'T' which stands for 'Task'.

SEE ALSO

int, shell

kill

Kill will delete a list of files from the system.

SYNTAX

```
kill [+dp] file ...
```

DESCRIPTION

Kill will delete or remove a list of file names from the system. If the file name killed was the last link to the file, the data will also be deleted, freeing up the space where it was residing. The '+p' option will cause kill to prompt for each file specified. If the response line to the prompt starts with a 'y', the file will be killed, otherwise it will remain. Kill will ignore files which are directories unless the '+d' option is specified and then only delete the directory if it is empty. Note that a directory is considered empty if its only two entries are '.' and '..' (which can not be killed directly). The caller must have write permission in the directory in which the file resides in order to kill it. Write permission in the file itself is not required! A few examples demonstrate its use.

```
kill lister junk
kill +p *
kill +d *
```

The first example will delete the files 'lister' and 'junk' in the current working directory. The second line will prompt for each file found in the working directory, and if 'y' is the response, the corresponding file will be deleted. The last example is very dangerous! It will delete all files in the current working directory including any empty directories (the +d option) without any prompts!

SEE ALSO

link

link

Create a new link to an existing file on the system.

SYNTAX

```
link file1 file2
```

DESCRIPTION

This command is used to create a new link to the existing 'file1' and give it the name 'file2'. File2 must not already exist or an error will result. A link to a file is simply a name in a directory which references a file. When the link is made, the data, owner, and permissions of the file are left unchanged. If someone changes the contents of the file they will also appear in the file referenced by the new link since link does not make a new copy of the file, it simply sets up another name which references the file. It should be noted that it is not permitted to link to a directory or to a file on another device. As an example:

```
link /usr/john/trees mytrees
```

will make a new entry in the current working directory called 'mytrees' which references the existing file 'trees' in the directory '/usr/john'.

SEE ALSO

kill

list

List the contents of a file or several files on the standard output.

SYNTAX

```
list file ...
```

DESCRIPTION

This command will list the contents of the file(s) specified on the standard output. If multiple file names are specified, they will be listed one following the other, as if they were one file. List without a file name will take its input from the standard input until an end of file is encountered. If a file name argument is encountered which consists of a single plus sign ('+'), the standard input will also be read. Several examples will demonstrate the use of list.

```
list test
list test junk >jest
list chp1 chp2 + chp3
```

The first line will list the contents of the file named 'test' on the standard output. The second example will list the contents of 'test' and 'junk' to the standard output. Note that the standard output has been changed to the file named 'jest' (see shell). The last example will list the file 'chp1' and 'chp2', then the standard input will be listed (the '+'), and finally the contents of 'chp3' will be listed. Note that list may be used to append files as in the second example above.

log

Log is the command used to terminate a user session or 'log off' of the system.

SYNTAX

```
log
```

DESCRIPTION

There are two methods of terminating an operating session (log off of the system). The first is to type a 'control d' character in response to the shell's prompt. The second is by use of the log command. Either method is acceptable. Log is part of the shell and does not reside on disk.

SEE ALSO

login, shell

login

Used to change from one user to another without 'logging off' of the system.

SYNTAX

```
login name
```

DESCRIPTION

The login command is used to change to a new user on a terminal. Typing login followed by the user name will cause the system to request the new user's password. If the password is entered correctly, the new user will be in command. Note that this is simply a time saver and is exactly the same as typing 'log' and then responding to the 'Login:' prompt. The login command is part of the shell.

SEE ALSO

log, shell

mail

The mail command is used to send mail to other users or to display ones own mail.

SYNTAX

```
mail
mail name ...
```

DESCRIPTION

The operating system supports a mail system which allows users to send messages to other users. In order to receive mail, it is necessary to create a file in your login directory called '.mail' (see 'create'). For security, this file should have its permissions set so others are unable to read or write it (see 'perms'). This file will normally not show up in a 'dir' listing since it starts with a period (see 'dir'). Each time you login, the system will check your '.mail' file, and if it is non-zero in size, the message 'You have mail.' will be displayed. To view your mail, simply type the command 'mail' without any arguments. Your mail will be displayed. Each entry will be preceeded by a header which gives the time and date the mail was received and who sent it. After the display, the prompt 'Save mail?' will appear. At this time you have three choices. Answering with a carriage return will leave the contents of the '.mail' file intact. Answering with 'n' will delete the contents, and a 'y' will transfer the mail to a new file called 'mailbox'. If mailbox did not exist, it will be created, otherwise the mail will be appended to the end of any existing text in the file. If either the 'y' or 'n' response was given, the contents of '.mail' will be deleted so that the system will not inform you of mail again until new mail is received.

The mail command can also be used to send mail to other users. You must know the login names of those you wish to send mail. Typing mail followed by the name or names of those you wish to send mail will cause the command to read information from the standard input until an end of file is seen (control d). At that time, the message will be sent to all names listed. Note that by using input redirection, the contents of a file may be sent as mail (see shell). A few examples will demonstrate the use of the mail command.

```
mail
mail john
mail john mary <letter
```

The first example would be used to display received mail. The second example would take all information typed up to a control d (end of file)

and send it to the user named john. The last example will send the contents of the file 'letter' to the users john and mary.

SEE ALSO

send, shell

makdev

The makdev command is used to create a new device type file on the system.

SYNTAX

```
/etc/makdev file type device unit
```

DESCRIPTION

The makdev command is restricted to use by the system manager and is used to create a new device name entry on the system. All devices look exactly like files on the system and therefore have names. The 'file' argument specifies the name for the device. The 'type' should be specified as a 'c' for character type devices or a 'b' for block type devices. The next argument, 'device', specifies the internal number which represents this device in the system, and the 'unit' argument specifies the unit number of the device specified. The System Configuration Guide should be consulted for details. Two examples follow:

```
/etc/makdev /dev/hdisk3 b 2 3
/etc/makdev /dev/ppunch c 5 0
```

The first line will create a device named '/dev/hdisk3', a block device with an internal device number of 2. This is unit number 3 of this device type. The second line will create a device named '/dev/ppunch' which is a character device, has a device number of 5, and is unit 0.

SEE ALSO

System Configuration Guide

Enable or disable the ability to receive messages from other users.

SYNTAX

```
message [yes] [no]
```

DESCRIPTION

The operating system supports a mechanism which allows users to directly communicate with other users currently logged on the system. The 'send' command is used for this purpose. The message command determines the ability to receive messages. Messages may be enabled or they may be locked out. For example:

```
message no
message yes
message
```

The first line will lock out all messages so others will not be able to 'send' to you. The second example will enable message reception. The last line will display the current message status without changing it.

SEE ALSO

send

mount

Mount a block type device on the system.

SYNTAX

```
/etc/mount device directory [r]
```

DESCRIPTION

The mount command may only be executed by the system manager and is used to mount a block device in the file systems directory structure. The 'device' name specified will be mounted at the node named 'directory'. After the mount, any references to 'directory' will reference the root directory of the device, and the previous contents of the directory will become inaccessible. If the optional 'r' argument is present, the device will be mounted as read only. Any attempts to write a device mounted read only will result in an error. As an example:

```
/etc/mount /dev/hdisk2 /usr2
```

will mount the device named '/dev/hdisk2' at the directory node named '/usr2'. Any future references to the directory '/usr2' will actually access the root directory of the device '/dev/hdisk2'.

SEE ALSO

unmount

Move a file or files to another location. The command may also be used to rename files.

SYNTAX

```
move file1 file2
move file ... directory
```

DESCRIPTION

The move command is used to move or change the name of files. The first form of the command will change the name of 'file1' to 'file2'. The user must have write permission in the directory the file resides. The second form of the command will move all of the files named into the 'directory' specified. All files will retain their original names. Note that the files are not copied unless an attempt to move a file across devices is made. A few examples will demonstrate its use.

```
move parts22 parts.old
move game /usr/john/mygame
move jobc1 jobc2 jobc3 /usr/john
```

The first example will move (rename) the file named 'parts22' to 'parts.old'. The second example will move (rename) the file 'game' to the file named 'mygame' in the directory '/usr/john'. The last case will move the list of files 'jobc1', 'jobc2', and 'jobc3' to the directory '/usr/john'. The files will retain their original name.

SEE ALSO

copy, rename

owner

The owner command is used to change the owner of a file.

SYNTAX

```
owner newowner file ...
```

DESCRIPTION

The owner command, which can only be run by the system manager, is used to change the owner of a file or list of files. The 'newowner' may either be a valid user's name or a user id number. Any number of files may be specified. As an example:

```
owner john /usr/john
```

will set the owner of the file '/usr/john' to be john.

page

Page format a file or list of files for display on a crt terminal or for output on a printer.

SYNTAX

```
page [+options] [file] ...
```

DESCRIPTION

The page command is used to format files for printing on a line printer or for viewing on a high speed crt terminal. Any number of files may be specified. If no files are specified, page will read the standard input. All output goes through the standard output which means the page command may be used in a pipe construct. The available options are as follows:

+l	preceed each line with a line number
+f	finish each page with line feeds instead of form feeds
+N	where N is a decimal number sets crt screen length
+pN	where N is a number sets the printer page length in lines

The +N option implies that the output will be viewed on a crt. After each N lines, the output will stop until an 'escape' character is typed. The only other formatting allowed with this option is +l which specifies line numbering. If +N is not specified, each page will be output with a header consisting of title (file name), date and time, and the page number. Normally each page is terminated with a form feed, but the +f option will substitute the correct number of line feeds to replace it. Page length defaults to 66 lines but may be set using the +p option. A few examples will demonstrate the use of page.

```
page text
page +124 junk
```

The first example will list the file 'text' with a header on each page. The second line will display the file 'junk' 24 lines at a time and preceed each line with a line number.

SEE ALSO

list

password

Set or change a user's login password.

SYNTAX

password [name]

DESCRIPTION

The password command is used to set or change a user's password. Only the system manager may set another user's password. When this command is invoked, the system will request the current password if it exists. If this is entered correctly, a prompt for the new password will be displayed. It should be noted that the characters typed for the password will not be echoed. After the new password is entered, the system will request the new password again, just to make sure there were no typos. If all goes well, the new password will be installed on the system and will have to be used for the next login. Passwords in general should be from 4 to 6 characters in length. A few examples will demonstrate the use of this command.

```
password
password john
```

The first example will allow a user to change his or her own password. The second example would only be permitted for the system manager, and will allow him to set the password for user 'john'. In this case, the old password is not requested.

path

Path is used to display the path name of the current working directory.

SYNTAX

path

DESCRIPTION

Typing path as a command will cause the system to print the complete path name of the current working directory. This is useful if one has been changing directories frequently and forgets the current location.

SEE ALSO

chd

perms

Set or change the permissions associated with a file or list of files.

SYNTAX

```
perms parameters file ...
```

DESCRIPTION

The perms command is used to set or change the permission bits associated with a file. There are six separate permissions associated with each file. The first three determine the permissions of the file's owner and the second three determine the permissions of all others. The three permissions in each group are read, write, and execute. Read permission will allow the file to be read or listed. Write permission will allow the file to be written or modified. Execute permission allows the file to be executed (see 'shell' for execution of non-binary files). Permissions for directories are the same except execute permission allows the directory to be 'searched' for another file name. If a directory does not have write permission granted, no new files can be created in it or deleted from it. The permissions specified in the 'perms' command are not subject to modification by the default permissions. The allowable 'parameters' are as follows:

u+[rwx]	set read, write, or execute permission for user (owner)
u-[rwx]	clear read, write, or execute for user (owner)
o+[rwx]	set permissions for others
o-[rwx]	clear permissions for others
s+	set user id bit
s-	clear user id bit

Any number of 'files' may be specified. Only the owner (or system manager) may change the permissions associated with a file. Following are several examples:

```
perms o-wx inven
perms u+x o+x lister
perms u-w o-rwx junk traps
```

The first example will turn off other's permissions for write and execute in the file 'inven'. The second example will set execute permission for the user (owner) as well as for others. The last line will disable write permission for the user and disable all permissions for others in the files 'junk' and 'traps'.

SEE ALSO

dir, dperm

print

Print a file or files on the system's printer.

SYNTAX

<print> [<file_name>] ... [+m]

Arguments

[<file_name>] The name of the file to print.

Options

m Suppress message to user.

DESCRIPTION

The <print> command may exist as several different names on the system. It is used to send a file or files to a printer spooler for printing. The <print> command usually has a name which represents the particular printer to be used for output. For example, if two printers are available, one might be called "lprinter" for line printer and the other, "daisy" for daisy wheel printer. Thus, the user has a choice of output devices. If no file is specified, <print> takes its input from standard input. The command can, therefore, be used in a pipe. Once the file has been sent for printing, the <print> command normally issues a message of the form:

"<file_name>" printed on <device> as "<user_name000>"

"<file_name>" is the name of the file printed (if available); <device> is the name of the printer device to which the file was routed; and the last item is the name which the system has assigned to the print file. This name, as well as the device name, is needed if it becomes necessary to delete the file from the print queue (see "purge"). This message may be inhibited by specifying the "+m" option on the command line.

SEE ALSO

insp
print control
purge

print

Print a file or files on the system's line printer.

SYNTAX

```
print file ...
```

DESCRIPTION

The print command may exist as several different names on the system. It is used to send a file or files to a printer spooler for printing. If no files are specified, the standard input is read, allowing print to be used in a pipe. The print command usually has a name which represents the particular printer to be used for output. As an example, if two printers are available, one might be called 'lprinter' for the line printer and 'daisy' for the daisy wheel printer. This gives the user a choice of output devices. Once the file has been sent for printing, the print command will issue a message similar to the following:

```
"file" printed on device as "username000"
```

The file is the name of the file printed (if available), device is the device name of the printer to which the file was routed, and the last item is the name which the system has assigned to the print file. This name as well as the device name will be needed if it becomes necessary to delete the file from the print queue (see purge).

SEE ALSO

insp, print control, purge

print control

A variety of commands to control the jobs being printed by a printer spooler.

SYNTAX

```
end spooler_device
idle spooler_device
next spooler_device
pstop spooler_device
rerun spooler_device
```

DESCRIPTION

These printer control commands are generally only available for use by the system manager. Each may only be run with an active spooler device.

End. This command is used to stop the currently printing job on the specified spooler device. The unprinted part of the file will be discarded, and the next job in the print queue will start printing.

Idle. The idle command will cause the spooler to become idle when it completes any job it may be currently printing. The 'next' command is used to restart the spooler from an idle state.

Next. This command is used to restart the specified spooler device when it is in the idle state. Either the 'idle' or 'rerun' command makes the spooler idle.

Pstop. Stop the specified spooler device. Any files in the queue will still be there the next time the spooler is initiated (using 'insp'). All active spoolers should be stopped before the system is shutdown. Printed data may be lost if this command is executed while a job is currently being printed.

Rerun. The rerun command will end the current job, place the file back on the print queue, and put the spooler in the idle state. The next command is used to restart the spooler.

SEE ALSO

insp, print, purge

prompt

Define the prompt and reprompt strings issued by the nshell program.

SYNTAX

```
prompt <prompt_str> [<reprompt_str>]
```

DESCRIPTION

The "prompt" command, which is part of the nshell program, defines the prompt and reprompt strings returned by the operating system. The prompt string is the string issued by the operating system when it is ready to accept a command. The reprompt string is the string issued by the operating system when the user has entered a line terminated by the combination of a backslash character and a carriage return. This combination of characters tells the nshell program that the line is incomplete.

Both the prompt and the reprompt strings should be enclosed in single or double quotation marks. Neither string may be more than fourteen characters long. The first tilde, '~', in each string is replaced in the prompt by the current time expressed in the form <hr>:<min>, which represents the hours and minutes on a 24-hour clock.

Arguments

<prompt_str>	The string to use as the new prompt. By default, the nshell program issues the following prompt: ++
<reprompt_str>	The string to use as the reprompt string, the prompt returned by the operating system when the user enters a line terminated by a backslash character followed by a carriage return. By default, the nshell program issues the following two-character reprompt: +>

EXAMPLES

```
1.  prompt '% '
2.  prompt '~ --> ' 'cont: '
```

The first example changes the prompt to a percent sign, followed by a space.

(continued)

The second example sets the prompt to the current time, followed by a space, followed by an arrow, followed by another space. It changes the reprompt string to "cont: ".

NOTES

- The "prompt" command is only effective while the nshell program under which it is invoked is running. The prompt and continuation prompt of the login nshell can be permanently altered by placing the appropriate command in the file ".startup" in the user's home directory. This file is automatically executed each time the user logs in.

SEE ALSO

nshell

prompt

Define the prompt string issued by the shell program.

SYNTAX

```
prompt [arg]
```

DESCRIPTION

This command is part of the shell and does not reside on disk. The standard shell prompt is '++ ' but may be changed at any time. The maximum allowed length of the prompt is 24 characters. The 'arg' should generally be enclosed in quotes. If the arg starts with a tilde, the prompt will start with the current time in the form HH:MM which represents hours and minutes (24 hour clock). Executing prompt without an argument will set the prompt back to the standard one. A few examples will demonstrate its use.

```
prompt '% '
prompt 'john: '
prompt '~> '
prompt
```

The first example will set the prompt to a percent sign followed by a space. The second line will set the prompt to the name john followed by a colon and a space. The third example will set the prompt to the time, followed by an angle bracket and a space. Finally, the last line will set the prompt back to the standard two plus signs.

SEE ALSO

shell

purge

Delete a file from a printer spooler queue.

SYNTAX

```
purge spooler_device file
```

DESCRIPTION

The purge command is used to delete a file from the specified spooler device queue. The file name is the name which 'print' reported when the file was printed (see print). If the file has already started printing, the purge command will have no effect. As an example:

```
purge lprinter john231
```

will delete the print job named 'john231' from the print queue for the device named 'lprinter'.

SEE ALSO

print, print control

qdb

Qdb is a quick debug program useful for assembly language program debugging.

SYNTAX

```
qdb [file]
```

DESCRIPTION

The qdb program is a small but useful debugging tool. It allows the examination of 'core' files. If the core 'file' name is not specified, 'core' is the default (in the current working directory). Qdb will prompt with a '?' as a command and allows the following commands.

```
n     display the command's calling line (name)
q     quit (exit qdb)
r     display the registers and their contents (in hex)
s     display memory starting at the current stack pointer
x     display 16 bytes of program memory starting at 'x' (hex)
x-x   display the program memory between the 2 addresses
```

The 'x' response should be a hex address in the program space. If an address is specified which was not part of the program an error will be reported ('bad address'). Specifying two addesses separated by a '-' will display memory between those addresses. Any other command input will display the first 16 bytes of the program address space.

rename

Rename is used to give a file another name.

SYNTAX

```
rename file1 file2
```

DESCRIPTION

Rename is used to give the file named 'file1' a new name called 'file2'. The new name must not already exist. This command is exactly like 'move file1 file2' and is just provided for convenience. As an example:

```
rename inv3-20 oldinv
```

will rename the file named 'inv3-20' to 'oldinv'. The file's contents will remain unchanged.

SEE ALSO

move

send

Send a message to another user on the system.

SYNTAX

send name

DESCRIPTION

The send command allows a user to directly communicate with another user. An error will be issued if the user you are sending to has locked out messages or is not currently logged on the system. After executing 'send', each line typed on the terminal (standard input) will be sent to the user specified. Since messages are sent one line at a time, it is possible for the receiver to 'send' back, thus creating a two way communications path. Typing a control D (end of file) will terminate the send operation and display the letters 'EOM' for end of message on the receivers terminal.

SEE ALSO

message

setpath

Display or redefine the names of the directories that the nshell program searches when looking for an executable file.

SYNTAX

setpath [<dir_name_list>]

DESCRIPTION

The "setpath" command, which is part of the nshell program, is used either to display or to redefine the list of names of the directories that the nshell program searches when looking for an executable file. This list is known as the search path. The nshell program searches the directories in the order in which the user specifies them on the command line.

Arguments

<dir_name_list> A list of the names of directories to use as the search path. The default list consists of the following directories: the user's working directory, "<home_dir>/bin", "/bin", "/usr/bin", and, if the user is the system manager, "/etc". (The home directory is the user's login directory, as specified in the password file.)

EXAMPLES

1. setpath . bin /bin /usr/games

This example sets the search path so that it consists of the user's working directory, the directory "bin" in the user's working directory, and the directories "/bin" and "/usr/games".

NOTES

- The "setpath" command is only effective while the nshell program under which it is invoked is running. The list of directories searched by the login nshell can be permanently altered by placing the appropriate command in the file ".startup" in the user's home directory. This file is automatically executed each time the user logs in.

(continued)

SEE ALSO

addpath
nshell

shell

The shell is the main system command language which is used to accept and execute user typed commands.

SYNTAX

```
shell [+bc] [args ...]
```

DESCRIPTION

The shell is the system's command language. All commands typed from the terminal are accepted by the shell, processed, and sent to the system for execution. This is the program which normally runs upon logging in to the system and greets you with the '++' prompt. Command lines accepted by the shell consist primarily of a command name followed by a list of arguments for the command. The first name on the command line is always assumed to be the name of a command. The following elements of the line are either passed to the command to be run, or processed by the shell as described below. All elements of the command line may be separated by either spaces or commas.

Since most of the commands reside on disk, the shell must locate the command name before it can be executed. The shell first looks in the current working directory for an executable file having the same name as the command specified. If found, the command is executed. If it is not found, the shell will prepend the string 'bin/' to the name and look again. This allows private binary files in your working directory. If the name is still not found, '/bin/' is prepended to the name. This causes shell to look for the command in the standard system binary directory '/bin' which is where most of the common commands are stored. If the command is still not located, one last attempt is made by prepending the string '/usr/bin' to the name. Lesser used commands are kept in this directory. If the command cannot be found, an error message will be issued.

The command name must be an executable file. There are three forms of files which may be safely set executable. The first is a standard binary file as produced by the assembler or compilers. The second is a text file which contains commands to be executed by the shell. The third is a BASIC compiled file. The shell recognizes these latter two as special cases, and in the first, a subshell is spawned to interpret the file of commands, and in the second, BASIC is loaded and run to execute the basic compiled file.

Multiple commands per line may be entered by separating the command strings with a semicolon. Commands entered in this fashion will be executed sequentially. Each command will be allowed to complete, then the next one on the line will be started. If an error termination from a command is detected, no further commands on the line will be executed. Multiple commands per line may also be separated with an ampersand (&). The ampersand acts exactly like the semicolon but the shell does not wait for the command to complete before executing the next one. The ampersand may be used to terminate a single command on a line as well, in which case the prompt will be issued immediately, not waiting for the command to finish. This allows the execution of commands in the background. When a command is started with the ampersand, the shell will report its task identification number preceeded by a 'T'. This number may be necessary for a subsequent 'wait' or 'int' operation. The task id is also reported as a result of executing the 'jobs' command.

Commands may also be separated by a vertical bar '|' or circumflex (or caret) '^'. This mechanism is called a pipe and allows the connection of several filter type programs. Two or more commands separated by '|' or '^' are connected such that the standard output of all but the last command is used as the standard input for the following command. All of the commands are effectively executed simultaneously.

Multiple commands may also be grouped using parentheses. This allows several commands to be executed sequentially and each having their output piped or redirected. Redirection of i/o is another capability of the shell. Normally commands get their input from the terminal (standard input) and place their output on the terminal (standard output). This i/o may be redirected to or from files by special shell arguments. An argument of the form '>string' causes the file named 'string' to be used as the standard output. If the file previously existed it will be truncated to zero length, else the file will be created. All output generated by the command and issued to file descriptor 1 (standard output) will be redirected to the specified file, and will not be displayed on the terminal. An argument of the form '>>string' is similar to the above with one exception. If the file specified by 'string' already exists, it will not be truncated, but instead, the new output data will be appended to the end of the file. Input redirection is also possible. Arguments of the form '<string' cause the associated command to get its input from the file specified instead of the terminal. The file is read as file descriptor 0 (standard input).

Several examples will demonstrate the various ways of presenting commands to the shell.

```
list text1 text2
dir; who
asmb taxes >taxes.out &
(dir; list inven) | page
date >infofile; who >>infofile
```

```
(date; who) >infofile
```

The first example is a simple command line. The command to be executed is 'list' and the arguments 'text1' and 'text2' will be passed to the list command. The second line shows multiple statement per line sequential command execution. Here, the 'dir' command would be executed, and when it completes, the 'who' command will be executed. The next example demonstrates output redirection and background command execution. This line will cause the shell to start the 'asmb' command running on the file 'taxes' which is passed as an argument. The output will not be displayed on the terminal but instead will be redirected to the file 'taxes.out'. The '&' tells the shell not to wait for the command to complete (background execution). The fourth example will pipe the standard output of 'dir' and 'list' of file inven (which are executed sequentially) into the standard input of the 'page' command. If the parentheses were not used, only the output of the list command would have gone through the pipe. The last two examples perform exactly the same. Here the output of the 'date' and 'who' commands will end up in the file 'infofile'.

Arguments are normally passed to the command as entered. There are two exceptions to this. The first is arguments containing name matching characters. The name matching characters are '*', '?', and '[]'. Arguments containing these characters are used as models for name matching in the current directory. The '*' character will match any string of characters in a file name, including the null string. Files starting with a leading period will be matched only if the leading period is specified in the match string. The '?' will match any single character in a file name (but not a null character). The '[]' construct will match any single character in the file name which is in the class of characters contained within the square brackets. The character class is defined by listing single characters or two characters separated by a '-' character within the brackets. Two characters separated by a hyphen defines a class of characters lexically between the pair. If the name matching characters appear in a path name, they must be in the last name of the path only. Several examples demonstrate name matching.

```
*
?
chapter[1-9]
[abcr-u]*.bak
/usr/john/a*.?
```

The first example will match all file names in the current directory. The second example will match all single character file names. The next example will match all file names starting with the string 'chapter' and ending with a digit between 1 and 9. The next line will match all file names starting with the characters 'a', 'b', 'c', or 'r' through 'u', and ending with the string '.bak'. The last example will match all file names in the directory '/usr/john' which start with 'a' and have a period as the next to last character in the name.

The second form of argument alteration by the shell involves quoting an argument. If an argument is needed which requires one of the special name matching characters or a space or comma in the argument, simply enclose the argument in single or double quotes. For example:

```
'This is a test'
"special*arg"
```

The first line would create one argument consisting of the string - This is a test. The second line will not do name matching with the '*' but instead will pass the argument as is directly to the command.

Command termination is normally not reported by the shell unless it terminates abnormally (because of a program interrupt). As an example, interrupting a program with a control C will cause the shell to print 'INTERRUPT' on the terminal to signify that the command did not terminate normally. Following is the complete list of abnormal termination messages:

```
Ended.
Hangup.
INTERRUPT!
Quit.
EMT trap.
Killed.
Pipe.
Argument.
Trace.
Time limit.
Alarm.
Terminated.
Interrupt #12.
```

Each of the above messages may have the string '(core dumped)' appended to it to signify the creation of a core file. Tasks run in the background will always report termination, even if terminating normally (Terminated.).

Logging off of the shell may be handled in two different ways. The first is by typing 'log' which simply causes the shell to terminate. The second is by typing an end of file (control D). This allows the shell to terminate if it is receiving input from a file instead of the terminal.

The shell also supports a variety of special commands. These include chd, dperm, jobs, log, login, prompt, and wait. They are covered separately in this manual.

Shell may be called with arguments and with two options. The options are as follows:

+b	execute the shell ignoring quits and interrupts
+c	execute the following argument (only argument) as a command line.

The arguments passed to the shell are available as $1 through $N where N is the last argument specified. The $0 specifies the name of the calling program.

SEE ALSO

chd, dperm, jobs, log, login, prompt, wait

shutup

Take the system from multi-user mode to single-user mode.

SYNTAX

```
/etc/shutup [[-]minutes]
```

Arguments

[minutes]	The number of minutes to wait before switching modes.
[-]	Omit hang-up interrupt and 15-second delay.

DESCRIPTION

"Shutup" takes the system from multi-user mode to single-user mode after the specified number of minutes. If no time is specified, the default is fifteen minutes. The maximum time specified should be sixty minutes. A time specification of zero minutes causes the system to switch modes immediately.

By default, the system is taken from multi-user mode by issuing a hang-up interrupt to all tasks, followed by a fifteen-second delay before actually entering single-user mode. If any tasks are still executing when "shutup" is run, the hang-up interrupt permits many of them to terminate cleanly without loss of data. If desired, the hang-up interrupt and fifteen-second delay can be omitted by specifying a hyphen, or minus sign, directly in front of the minutes specification.

Only the system manager may execute this command. The task number of the "shutup" program is reported when invoked so that it can be terminated using "int" if necessary. "Shutup" sends a message to all terminals that are logged in at various time intervals to warn users of the impending system shutdown.

EXAMPLES

```
shutup 30
```

This command causes the system to enter single-user mode thirty minutes after it is issued.

SEE ALSO

int
stop

shutup

The shutup command is used to take the system from mult-user mode back to single-user mode.

SYNTAX

```
/etc/shutup [minutes]
```

DESCRIPTION

Shutup will take the system from multi-user mode to single-user mode after the specified number of minutes. If no time is specified, 15 minutes will be the default. The maximum time specified should be 60 minutes. A time specification of 0 will shut the system down immediately. Only the system manager may execute this call. The task number of the shutup program is reported when invoked so it may be terminated using 'int' if necessary. Shutup sends a message to all logged on terminals at various time intervals to warn users of system shutdown. As an example:

```
shutup 30
```

will cause the system to enter single-user mode in 30 minutes.

SEE ALSO

int, stop

Bring the system to a grinding halt.

SYNTAX

/etc/stop

DESCRIPTION

The stop command is used to halt the system. It may only be executed by the system manager and while in single-user mode. Use 'shutup' to exit multi-user mode. This is the only safe way to shut down the system.

SEE ALSO

shutup

ttyset

Examine or adjust the parameters associated with the user's terminal.

SYNTAX

```
ttyset [<param_list>] [+]
```

DESCRIPTION

The "ttyset" command displays or changes the configuration of various system parameters relating to the user's terminal. The default configuration for all terminals is as follows:

-raw	+echo	+tabs	-case
+alf	+becho	-schar	-cntrl
+esc	-xon	-any	
8 Data	1 Stop	none	
crt	bs=08	dl=18	

The default baud-rate is system dependent.

Arguments

<param_list>	The list of parameters to adjust. If the user specifies no parameters, the command writes the current configuration of the user's terminal to standard output.

Format for Arguments

+raw	Set raw I/O mode. This mode is normally used only by special programs.
+echo	Enable character echo.
+tabs	Expand tabs on output. This parameter should be set for terminals which do not perform tab expansion, so that the system expands tabs to every eighth column.
+case	Convert upper- to lowercase on input and lower- to uppercase on output.
+alf	Output a line-feed character after each carriage return.
+becho	Enable echo of backspace character. When enabled, this parameter outputs an extra backspace and space character to erase on-screen characters for those terminals which do not automatically do so.

(continued)

+schar	Accept input one character at a time. This mode is normally used only by special programs.
+cntrl	Ignore meaningless control characters (all control characters except carriage return, horizontal tab, control-C, control-D, control-backslash, backspace, and the line-delete character). When enabled, this parameter keeps strange, unwanted characters out of the user's files.
+esc	Enable the escape character (hexadecimal 1B) for starting and stopping output.
+xon	Enable the XON/XOFF mechanism for starting and stopping output. Control-S stops output; control-Q starts it. When output is stopped, XOFF characters are ignored; when output is not stopped, XON characters are ignored.
+any	Accept any character to start output after it has been stopped by either an escape character or a control-S.
bs=<num>	Set the backspace character to hexadecimal <num>. Default is hexadecimal 08 (control-H).
dl=<num>	Set the line-delete character to hexadecimal <num>. Default is hexadecimal 18 (control-X).

All parameters beginning with a plus sign, '+', may instead begin with a minus sign, '-', to set the opposite condition.

In addition, depending on the hardware, the user may be able to set the following parameters:

db=<7_or_8>	Set the number of data bits.
sb=<1_or_2>	Set the number of stop bits.
even	Even parity mode.
odd	Odd parity mode.
none	No parity.
b=<num>	Set the baud rate to <num>. Legal values of <num> are 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, and 19200.

Only certain combinations of values for data bits, stop bits, and parity are legal. The legal configurations are shown in Table 1.

(continued)

Table 1. Legal Configurations for Data Bits, Stop Bits, and Parity

Data Bits	Stop Bits	Parity
7	2	Even
7	2	Odd
7	1	Even
7	1	Odd
8	2	None
8	1	None
8	1	Even
8	1	Odd

Some hardware configurations allow the user to set the following delays for mechanical terminals which require time for long carriage movements, such as carriage returns. Experimenting with the various delays determines the correct setting for a particular terminal.

crt	Set delays for crt (no delays).
hcs	Set slow hard-copy delays.
hcm	Set medium hard-copy delays.
hcf	Set fast hard-copy delays.

Options Available

+ Display the current values for all "ttyset" parameters after making the changes.

EXAMPLES

1. ttyset
2. ttyset +alf -becho
3. ttyset +tabs hcm +

The first example displays the current values of the "ttyset" parameters for the user's terminal.

The second example enables the automatic output of a line-feed character after a carriage return and disables the echo of the backspace character.

The third example enables the expansion of tab characters on output and sets a medium speed, hard-copy delay. The command displays the current "ttyset" parameters after making the changes.

(continued)

NOTES

- The "ttyset" command is generally used to set the configuration of the terminal from which it is called. Assuming permissions are granted, it is possible to execute the "ttyset" command on another terminal or even a serial printer device by redirecting input from the desired device. For example, the following command enables XON/XOFF processing on the serial printer "spr":

```
ttyset +xon </dev/spr
```

  Although the "ttyset" command can be performed on a serial printer device, the only argument which can be set or which has any effect is the "xon" argument.

- The "ttyset" parameters "raw", "schar", and "tabs" are reset by the shell program each time it issues a prompt for a command. Thus, if the user types the command "ttyset +raw" in response to a prompt from the shell program, raw I/O mode is enabled, but when the command is complete, the shell program turns raw I/O mode off before issuing the next prompt. To the casual observer it appears that the "ttyset" command failed. If, however, the same command is performed by an "exec" statement in a BASIC program, raw I/O mode remains in effect for the duration of the BASIC program (unless specifically turned off by another "ttyset" operation) because the shell program is not reentered.

- The parameters which set data and stop bits, baud rate, and parity can only change when the terminal goes from a closed to an open state. Thus, they can easily be set by a command in the startup file "/etc/startup". However, in order to set them after the terminal has been opened, the user must issue the appropriate "ttyset" command, log out, and log in again.

  The baud rate can also be set by editing the file "/etc/ttylist". Each entry for an active terminal consists of a plus sign, "+", followed by a space, followed by the two-digit number that represents that particular terminal. The user can change the baud rate of a given terminal by replacing the space in the corresponding entry with a single hexadecimal digit. Each digit corresponds to a particular baud rate (see Table 2). Each time the system sends a login prompt to an active terminal, it sets the baud rate for that terminal according to the rate specified by this digit. Thus, a baud rate set in the file "/etc/ttylist" always overrides an attempt to change the baud rate with the "ttyset" command.

(continued)

Table 2. Correspondence between Hexadecimal Digits and Baud Rates

Digit	Baud Rate	Digit	Baud Rate	Digit	Baud Rate
1	75	6	300	B	3600
2	110	7	600	C	4800
3	134.5	8	1200	D	7200
4	150	9	1800	E	9600
5	nd	A	2400	F	19200

Note: nd = not defined

ERROR MESSAGES

Error getting terminal status: <reason>
The operating system returned an error when "ttyset" tried to get information on the current configuration of the terminal. This message is followed by an interpretation of the error returned by the operating system.

Error setting terminal characteristics: <reason>
The operating system returned an error when "ttyset" tried set the new parameters. This message is followed by an interpretation of the error returned by the operating system.

Illegal baud rate specified: <num>
Command aborted!
Legal values for the baud rate are 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, and 19200.

Illegal configuration: '# Data bits, # Stop bits, <set> Parity'.
The combination of values used to set data bits, stop bits, and parity is illegal. See Table 1 for a list of the legal combinations.

Invalid parameter, "<param>", found.
Command aborted!
The parameter specified by <param> is not a valid parameter for the "ttyset" command.

ttyset

Examine or adjust the parameters associated with the user's terminal.

SYNTAX

```
ttyset [arguments]
```

DESCRIPTION

The ttyset command allows configuration of various system parameters relating to the user's terminal. Entering the command without any arguments will display the current terminal configuration. The allowed arguments are as follows:

+raw	set raw mode
+echo	enable character echo
+tabs	expand tabs on output
+case	convert upper to lower case on input and vice versa
+becho	enable backspace echo
+alf	output line feed after each carriage return
+schar	single character input mode
+cntrl	ignore meaningless control characters
crt	crt delays (no delays)
hcs	slow hard copy delays
hcm	medium hard copy delays
hcf	fast hard copy delays
bs=xx	define backspace character as hex xx
dl=xx	define line cancel character as hex xx

Note that all arguments shown beginning with '+' may be started with '-' instead to set the opposite condition. As an example, '-echo' would turn off character echo on input. Raw mode and schar mode are normally only used by special programs. Terminals which do not perform tab character expansion should set '+tabs' to have the system expand tabs to every 8th column. The becho option will output an extra backspace and space character to erase on screen characters for those crts which do not automatically. The cntrl mode will cause the system to ignore all control characters not having special meaning. This is useful to keep strange, unwanted characters out of files.

The delays are used for mechanical type terminals which require time for long carriage movements (such as carriage returns). Experimenting with the various delays will determine the correct setting for your terminal. The default backspace and line cancel characters are control H (hex 08) and control X (hex 18) respectively. These may be changed with ttyset.

Several examples will demonstrate the use of this command.

```
ttyset
ttyset +alf -becho
ttyset +tabs hcm +
```

The first example will simply display the current values of the ttyset parameters. The second line will enable the automatic line feed and disable backspace echo. The last example will enable tab expansion on output and set the delays for a medium speed hard copy terminal. The lone '+' in this example will cause the new ttyset parameter values to be displayed after the changes have been made.

udate

Display or set the date in Universal Coordinated Time (GMT).

SYNTAX

udate [date_specification]

DESCRIPTION

The udate command will display the current date and time in GMT rather than local time. The date specification for setting the date and time is the same as in the 'date' command. This command is provided primarily because of the one obscure hour immediately following the change from US Daylight Savings Time to standard time which is unsettable by 'date'.

SEE ALSO

date

unmount

Unmount a previously mounted device from the file system.

SYNTAX

```
/etc/unmount device
```

DESCRIPTION

Unmount the device specified from the file system. If the device was not previously mounted, an error will result. Only the system manager may execute this command. An example demonstrates its use.

```
/etc/unmount /dev/fd1
```

This will unmount the device named '/dev/fd1'. The directory on which it was mounted will again be usable.

SEE ALSO

mount

DESCRIPTION

Wait for a background task to complete.

SYNTAX

wait [taskid]

DESCRIPTION

The wait command is part of the shell and is used to wait for the completion of a task which was initiated as a background task (using the & in shell). The system will 'wait' for all active background tasks to complete before finishing unless a taskid is specified. The termination status of the task will be reported when it completes. A control C may be used to interrupt the waiting if desired. As examples:

```
wait
wait 3276
```

The first example will wait for all background tasks to complete while the second one will wait only until task number 3276 finishes.

SEE ALSO

shell

who

Report who is currently on the system.

SYNTAX

```
who
```

DESCRIPTION

The who command is used to get a report of the users currently on the system. The information displayed consists of the users login name, terminal name (as the system knows it), and the login time.

who

Report who is currently on the system.

SYNTAX

```
who [am i]
```

DESCRIPTION

The "who" command is used to obtain a report of all users currently on the system. The information displayed for each user consists of the user name, the terminal being used (as the system knows it), and the time the user logged in.

If the optional "am i" parameter follows the who command, only the name of the user executing the command is displayed.