The Open Group Base Specifications Issue 7, 2018 edition
IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)

12. Utility Conventions

12.1 Utility Argument Syntax

This section describes the argument syntax of the standard utilities and introduces terminology used throughout POSIX.1-2017 for describing the arguments processed by the utilities.

Within POSIX.1-2017, a special notation is used for describing the syntax of a utility's arguments. Unless otherwise noted, all utility descriptions use this notation, which is illustrated by this example (see XCU *Simple Commands*):

```
utility_name[-a][-b][-c option_argument]
    [-d|-e][-f[option_argument]][operand...]
```

The notation used for the SYNOPSIS sections imposes requirements on the implementors of the standard utilities and provides a simple reference for the application developer or system user.

1. The utility in the example is named *utility_name*. It is followed by options, option-arguments, and operands. The arguments that consist of <hyphen-minus> characters and single letters or digits, such as `'a'`, are known as "options" (or, historically, "flags"). Certain options are followed by an "option-argument", as shown with [**-c** *option_argument*]. The arguments following the last options and option-arguments are named "operands".

2. Option-arguments are shown separated from their options by <blank> characters, except when the option-argument is enclosed in the `'['` and `']'` notation to indicate that it is optional. This reflects the situation in which an optional option-argument (if present) is included within the same argument string as the option; for a mandatory option-argument, it is the next argument. The Utility Syntax Guidelines in Utility Syntax Guidelines require that the option be a separate argument from its option-argument and that option-arguments not be optional, but there are some exceptions in POSIX.1-2017 to ensure continued operation of historical applications:

    a. If the SYNOPSIS of a standard utility shows an option with a mandatory option-argument (as with [**-c** *option_argument*] in the example), a conforming application shall use separate arguments for that option and its option-argument. However, a conforming implementation shall also permit applications to specify the option and option-argument in the same argument string without intervening <blank> characters.

    b. If the SYNOPSIS shows an optional option-argument (as with [**-f**[*option_argument*]] in the example), a conforming application shall place any option-argument for that option directly adjacent to the option in the same argument string, without intervening <blank> characters. If the utility receives an argument containing only the option, it shall behave as specified in its description for an omitted option-argument; it shall not treat the next argument (if any) as the option-argument for that option.

3. Options are usually listed in alphabetical order unless this would make the utility description more confusing. There are no implied relationships between the options based upon the order in which they appear, unless otherwise stated in the OPTIONS section, or unless the exception in Guideline 11 of Utility Syntax Guidelines applies. If an option that does not have option-arguments is repeated, the results are undefined, unless otherwise stated.

4. Frequently, names of parameters that require substitution by actual values are shown with embedded <underscore> characters. Alternatively, parameters are shown as follows:

<parameter name>

The angle brackets are used for the symbolic grouping of a phrase representing a single parameter and conforming applications shall not include them in data submitted to the utility.

5. When a utility has only a few permissible options, they are sometimes shown individually, as in the example. Utilities with many flags generally show all of the individual flags (that do not take option-arguments) grouped, as in:

   `utility_name [-abcDxyz][-p` *arg*`][`*operand*`]`

   Utilities with very complex arguments may be shown as follows:

   `utility_name [options][`*operands*`]`

6. Unless otherwise specified, whenever an operand or option-argument is, or contains, a numeric value:

   - The number is interpreted as a decimal integer.
   - Numerals in the range 0 to 2147483647 are syntactically recognized as numeric values.
   - When the utility description states that it accepts negative numbers as operands or option-arguments, numerals in the range -2147483647 to 2147483647 are syntactically recognized as numeric values.
   - When the utility description states that the number is a file size-related value (such as a file size or offset, line number, or block count), numerals in the range 0 to the maximum file size supported by the implementation are syntactically recognized as numeric values (see XCU *Considerations for Utilities in Support of Files of Arbitrary Size*). Where negative values are permitted, any value in the range -(maximum file size) to the maximum file size is accepted.
   - Ranges greater than those listed here are allowed.

   This does not mean that all numbers within the allowable range are necessarily semantically correct. A standard utility that accepts an option-argument or operand that is to be interpreted as a number, and for which a range of values smaller than that shown above is permitted by the POSIX.1-2017, describes that smaller range along with the description of the option-argument or operand. If an error is generated, the utility's diagnostic message shall indicate that the value is out of the supported range, not that it is syntactically incorrect.

7. Arguments or option-arguments enclosed in the '[' and ']' notation are optional and can be omitted. Conforming applications shall not include the '[' and ']' symbols in data submitted to the utility.

8. Arguments separated by the '|' (<vertical-line>) bar notation are mutually-exclusive. Conforming applications shall not include the '|' symbol in data submitted to the utility. Alternatively, mutually-exclusive options and operands may be listed with multiple synopsis lines.

   For example:

   `utility_name -d[-a][-c` *option_argument*`][`*operand*`...]`
   `utility_name[-a][-b][`*operand*`...]`

   When multiple synopsis lines are given for a utility, it is an indication that the utility has mutually-exclusive arguments. These mutually-exclusive arguments alter the functionality of the utility so that only certain other arguments are valid in combination with one of the mutually-exclusive arguments. Only one of the mutually-exclusive arguments is allowed for invocation of the utility. Unless otherwise stated in an accompanying OPTIONS section, the relationships between arguments depicted in the SYNOPSIS sections are mandatory requirements placed on conforming applications. The use of conflicting mutually-exclusive arguments produces undefined results, unless a utility description specifies otherwise. When an option is shown without the '[' and ']' brackets, it means that option

is required for that version of the SYNOPSIS. However, it is not required to be the first argument, as shown in the example above, unless otherwise stated.

9. Ellipses ("...") are used to denote that one or more occurrences of an operand are allowed. When an option or an operand followed by ellipses is enclosed in brackets, zero or more options or operands can be specified. The form:

   `utility_name [-g option_argument]...[operand...]`

   indicates that multiple occurrences of the option and its option-argument preceding the ellipses are valid, with semantics as indicated in the OPTIONS section of the utility. (See also Guideline 11 in Utility Syntax Guidelines .)

   The form:

   `utility_name -f option_argument [-f option_argument]... [operand...]`

   indicates that the **-f** option is required to appear at least once and may appear multiple times.

10. When the synopsis line is too long to be printed on a single line in the Shell and Utilities volume of POSIX.1-2017, the indented lines following the initial line are continuation lines. An actual use of the command would appear on a single logical line.

12.2 Utility Syntax Guidelines

The following guidelines are established for the naming of utilities and for the specification of options, option-arguments, and operands. The *getopt*() function in the System Interfaces volume of POSIX.1-2017 assists utilities in handling options and operands that conform to these guidelines.

Operands and option-arguments can contain characters not specified in the portable character set.

The guidelines are intended to provide guidance to the authors of future utilities, such as those written specific to a local system or that are components of a larger application. Some of the standard utilities do not conform to all of these guidelines; in those cases, the OPTIONS sections describe the deviations.

Guideline 1:
Utility names should be between two and nine characters, inclusive.

Guideline 2:
Utility names should include lowercase letters (the **lower** character classification) and digits only from the portable character set.

Guideline 3:
Each option name should be a single alphanumeric character (the **alnum** character classification) from the portable character set. The **-W** (capital-W) option shall be reserved for vendor options.

Multi-digit options should not be allowed.

Guideline 4:
All options should be preceded by the '-' delimiter character.

Guideline 5:
One or more options without option-arguments, followed by at most one option that takes an option-argument, should be accepted when grouped behind one '-' delimiter.

Guideline 6:
Each option and option-argument should be a separate argument, except as noted in Utility Argument Syntax, item (2).

Guideline 7:
Option-arguments should not be optional.

Guideline 8:
When multiple option-arguments are specified to follow a single option, they should be presented as a single argument, using <comma> characters within that argument or <blank> characters within that argument to separate them.

Guideline 9:
All options should precede operands on the command line.

Guideline 10:

The first **--** argument that is not an option-argument should be accepted as a delimiter indicating the end of options. Any following arguments should be treated as operands, even if they begin with the '-' character.

Guideline 11:

The order of different options relative to one another should not matter, unless the options are documented as mutually-exclusive and such an option is documented to override any incompatible options preceding it. If an option that has option-arguments is repeated, the option and option-argument combinations should be interpreted in the order specified on the command line.

Guideline 12:

The order of operands may matter and position-related interpretations should be determined on a utility-specific basis.

Guideline 13:

For utilities that use operands to represent files to be opened for either reading or writing, the '-' operand should be used to mean only standard input (or standard output when it is clear from context that an output file is being specified) or a file named **-**.

Guideline 14:

If an argument can be identified according to Guidelines 3 through 10 as an option, or as a group of options without option-arguments behind one '-' delimiter, then it should be treated as such.

The utilities in the Shell and Utilities volume of POSIX.1-2017 that claim conformance to these guidelines shall conform completely to these guidelines as if these guidelines contained the term "shall" instead of "should". On some implementations, the utilities accept usage in violation of these guidelines for backwards-compatibility as well as accepting the required form.

Where a utility described in the Shell and Utilities volume of POSIX.1-2017 as conforming to these guidelines is required to accept, or not to accept, the operand '-' to mean standard input or output, this usage is explained in the OPERANDS section. Otherwise, if such a utility uses operands to represent files, it is implementation-defined whether the operand '-' stands for standard input (or standard output), or for a file named **-**.

It is recommended that all future utilities and applications use these guidelines to enhance user portability. The fact that some historical utilities could not be changed (to avoid breaking existing applications) should not deter this future goal.

return to top of page

[Main Index | XBD | XSH | XCU | XRAT]

<<< Previous Home Next >>>