POSIX.2 9

Portable Operating System Interface–Part 2: Shell and Utilities (POSIX.2)

In 1992, IEEE Standard 1003.2–1992 became part of the POSIX series of standards. Referred to as "POSIX.2," IEEE Standard 1003.2–1992 defines a standard interface and environment for applications that require a shell command language interpreter and a set of common utility programs. POSIX.2 is complimentary to ISO/IEC 9945–1:1990.

IEEE Std 1003.2-1992 also supplements the application portability interfaces to promote the "portability" of users and programmers between conforming systems. The User Portability Utilities Option extends the list of utilities, and features of utilities used primarily for application portability, to provide a common interactive environment.

This standard, referred to as "POSIX.2," is based upon documentation and the knowledge of existing programs that assume an interface and architecture similar to that described by POSIX.1. Within this chapter the POSIX.2 standard is referred to in places as "the standard."

Scope

To comply with section 1.3.1.2 (*Documentation*), this document describes the behavior of features in the SunOS 5.5 operating system which are described in the standard as *implementation-defined* or for which it is stated that implementations may vary. It does not describe any extensions or enhancements outside the scope of the standard.

9

Note – Section 2.2.1.5 of the standard defines the term *implementation-defined* as follows: "An indication that the implementation provider shall define and document the requirements for correct program constructs and correct data of a value or behavior."

The information contained within this chapter does not replace the POSIX.2 standard; rather, it serves as an adjunct to the standard for supplying the technical information needed by application developers to write source code within the SunOS 5.5 operating system framework.

Audience

This explication is for the experienced C programmer who, when writing an applications program designed to conform to the standard, needs to know the specific behavior of the *implementation-defined* features mentioned in the standard.

Each subsection is prefaced by the appropriate section taken directly from the standard and has the corresponding section number attached to the title.

Note – For maximum portability, applications should not depend upon any particular behavior that is *implementation-defined*.

Notation Used in the Remainder of this Chapter

The following format is used to identify which passage of text is quoted from the standard and which passage of text describes how the SunOS 5.5 operating system implements that area.

- **P.** stands for POSIX.
- **S.** stands for SunOS 5.5.

Section numbers cited in the remainder of this chapter correspond to those of the standard. When creating an application program, this format will help you to quickly locate additional information that you need from the standard.

9

Implementation-defined Areas of POSIX.2

POSIX.2 Section 1, General

1.3.1 Implementation Conformance

1.3.1.1 Requirements
P. The system may provide additional or enhanced utilities, functions, or facilities not required by this standard. Nonstandard extensions should be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of utilities, functions, or facilities defined by this standard.

S. The SunOS 5.5 reference manuals describe the utilities, functions, and facilities provided by the SunOS 5.5 operating system.

P. The conformance document shall define an execution environment in which an application can be run with the behavior specified by the standard.

S. When Solaris 2.5 is installed with SPARCompiler 4.0 on a SPARC based platform, or with ProCompiler™ C 3.0 and patch 102486-02 on an x86 based platform, and the application searches for standard utilities in the directories specified by the output of the `getconf PATH` command in the order specified from beginning to end, the utilities will behave as specified by POSIX.2.

1.3.1.2 Documentation
P. The conformance document shall contain a statement that indicates the full name, number, and date of the standard that applies.

S. IEEE Std 1003.2-1992 and IEEE Std 1003.2a-1992 are supported.

P. The conformance document may also list software standards approved by ISO/IEC or any ISO/IEC member body that are available for use by a Conforming POSIX.2 Application.

S. The available software depends on which packages have been installed.

9

POSIX.2 Section 2, Terminology and General Requirements

2.2.2 General Terms

2.2.2.8: appropriate privileges

P. An *implementation-defined* means of associating privileges with a process with regard to the function calls and function-call options defined in POSIX.1 that need special privileges. There may be zero or more such means.

S. The following is a list of references to "appropriate privileges" in the POSIX.2 standard:

3.5.3 Variables See section 3.5.3 in this document.

4.7.2 Description {of `chmod`} See POSIX.1 section 5.6.4.2 *Description {of Change File Modes*}.

4.30.2 Description {of `id`} See section 4.30.2 *Description {of* `id`} in this document.

4.35.2 Description {of `localedef`} See section 4.35.2 *Description {of* `localedef`*}* in this document.

4.35.4 Operands {of `localedef`} See section 4.35.4 *Operands* {*of* `localedef`} See section 4.35.4 Operands {of localedef} in this document.

5.17.2 Description {of `mesg`} See section *5.17.2 Description* {*of* `mesg`} in this document.

5.20.2 Description {of `nice`} See section 5.20.2 *Description* {*of* `nice`} in this document.

5.20.3 Options {of `nice`} See section 5.20.3 *Options* {*of* `nice`} in this document.

5.24.2 Description {of `renice`} See section 5.24.2 *Description* {*of* `renice`} in this document.

5.24.3 Options {of `renice`} See section *5.24.3 Options* {*of* `renice`} in this document.

5.28.2 Description {of `talk`} See section 5.28.2 *Description* {*of* `talk`} in this document.

5.37.2 Description {of `write`} See section 5.37.2 *Description* {*of* `write`} in this document.

2.2.2.27: byte
P. A byte is composed of a contiguous sequence of bits, the number of which is *implementation-defined*.

S. On Solaris 2.5, a byte is composed of 8 bits.

2.2.2.61: extended security controls
P. The access control and privilege mechanisms have been defined to allow *implementation-defined* extended security controls.

S. Access Control Lists are drfined in *SunOS Reference Manual* in the getfacl(1) man page.

2.2.2.65: file
P. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, and directory. Other types of files may be defined by the implementation.

S. Symbolic links are also defined.

2.2.2.68: file group class
P. A process is in the file group class of a file if the process is not in the file owner class and if the effective group ID or one of the supplementary group IDs of the process matches the group ID associated with the file. Other members of the class may be *implementation-defined*.

S. There are no additional file class members.

2.2.2.93: job control
P. POSIX.1 conforming implementations may optionally support job control facilities.

S. Job control is supported

2.2.2.120: parent process ID
P. After the lifetime of the creator has ended, the parent process ID is the process ID of an *implementation-defined* system process.

S. If a child process continues to exist after its creator process ceases to exist, the child process is inherited by `init`. The `init` process ID is 1.

2.2.2.121: pathname
P. A pathname that begins with two successive slashes may be interpreted in an *implementation-defined* manner.

S. Same as one slash.

2.2.2.141: read only file system
P. A file system that has *implementation-defined* characteristics restricting modifications.

S. A read-only file system does not allow for modification of its files or directories.

2.4 Character Set

P. Use of a locking-shift encoding with any of the standard utilities or the optional C-language functions that describe character (versus byte) or text-file manipulation is *implementation-defined*.

S. Use of locking-shift encodings is not supported.

2.4.1 Character Set Description File

P. It is *implementation-defined* whether or not users or applications can provide additional character set description files.

S. Additional character set description files are allowed.

P. Implementations supporting other byte sizes may allow constants to represent values larger than those that can be represented in 8-bit bytes, and to allow additional digits in constants. The manner in which these constants are represented in the character stored in the system is *implementation-defined*.

S. Bit sizes other than 8 bits are not supported.

2.5 Locale

P. Locales other than those supplied by the implementation can be created via the `localedef` utility provided that the {POSIX2_LOCALEDEF} symbol is defined on the system. Otherwise only the implementation provided locales(s) can be used.

S. The `localedef` utility is provided. Other locales can be created using this utility.

P. When the value of a locale environment variable begins with a slash (/), it shall be interpreted as the pathname of the locale definition; the type of file (regular, directory, etc.) used to store the locale definition is *implementation-defined*. If the locale value does not begin with a slash, the mechanism used to locate the locale is *implementation-defined*.

S. The mechanism to locate the locale is defined by the implementation of the `setlocale()` system call provided by Solaris 2.5. On Solaris 2.5, `setlocale()` supports composite locale names.

2.5.2: Locale Definition

P. If the file contains source definitions for more than one category, *implementation-defined* categories, if present, shall appear after the categories defined by this clause.

S. No additional categories or any additional keywords in any locale category source definition are defined.

P. (2) A character can be represented by the character itself, in which case the value of the character is *implementation-defined*.

S. All three supported locales, C, POSIX, and en_US, use the ISO8859-1 character encoding values.

2.5.2.1 LC_CTYPE

P. When the implementation automatically includes a missing character, it shall have an encoded value dependent on the *charmap* file in effect; otherwise, it shall have a value derived from an *implementation-defined* character mapping.

S. The character mapping used is the ISO8859-1 codeset.

2.5.2.5 LC_TIME

P. It is *implementation-defined* whether the following optional keywords shall be recognized.

era
era_year
era_d_fmt
alt_digits

S. These keywords are recognized.

2.5.3 Locale Definition Grammar

P. Any grammars for additional categories and keywords are *implementation-defined*.

S. Under the LC_CTYPE category we provide the `cswidth` keyword that specifies the size in bytes of the codesets. The Solaris internal prepresentation uses EUC which has four codesets.

The syntax of `cswidth` is:

```
cswidth    <n1>:<d1>, <n2>:<d2>, <n3>:<d3>
```

where <n?> is the number of byt4es for the codeset and <d?> is the number of display positions for the codeset for codesets 1, 2, and 3 respectively. Codeset 0 is always 1 byte.

LC_TIME recognizes the `date_fmt` keyword.

The syntax is:

```
date_fmt   "date format"
```

This value is accessed when a `%C` is sent to the date/time code.

2.6 Environment Variables

LANG
LC_COLLATE
LC_TYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME

P. Additional semantics of (these) variable(s), if any, are *implementation-defined*.

S. *The POSIX1003.2 Shell and Utility Application Interface* defines no additional semantics of these environment variables.

PATH
P. If **PATH** is unset or is set to null, the path search is *implementation-defined*.

S. When `PATH` is not set, Solaris supplies the default path `/usr/bin`.

P. If the **LANG** variable is not set or is set to the empty string, the *implementation-defined* default locale shall be used.

S. On Solaris 2.5, the implementation-defined default locale is "C".

P. If **LANG** (or any of the **LC_*** environment variables) contains one of a set of *implementation-defined* values, the standard utilities shall behave in accordance with the rules in a corresponding *implementation-defined* locale description for the associated category.

S. This is true for all locales that Sun supports.

P. Additional criteria for determining a valid locale name are *implementation-defined*.

S. None.

2.9.1.5 File Removal

P. When a directory that is the root directory or current working directory of any process is removed, the effect is *implementation-defined*.

S. If the directory indicated in the call to `rmdir()` is a mount point for a mounted filesystem, `rmdir()` sets `errno` to [EBUSY] and returns -1.

2.11.5.2 Input Files

P. *Implementations shall define* in the conformance documentation those utilities that are limited by constraints other than file system space, available memory, and other limits specifically cited by this standard; and identify what the constraint is; and indicate a way of estimating when the constraint would be reached.

S. The `cu`, `uucp`, `uux`, `uuname`, `uulog`, `uupick`, and `uuto` configuration files include:

```
/etc/uucp/Config   /etc/uucp/Dialers  /etc/uucp/Poll
/etc/uucp/Devconfig/etc/uucp/Grades   /etc/uucp/Sysfiles
/etc/uucp/Devices  /etc/uucp/Limits   /etc/uucp/Systems
/etc/uucp/Dialcodes/etc/uucp/Permissions
```

These file must be edited or otherwise manipulated in the POSIX locale. The behavior is undefined if multibyte characters are present in the configuration files.

P. Similarly, some utilities descend the directory tree (recursively). Implementations also shall document any limits that they may have in descending the directory tree that are beyond the limits cited by this standard.

S. Except for `du`, the *POSIX 1003.2 Shell and Utility Application Interface* meets the performance limits of the POSIX.2 standard. The depth of the hierarchy which can be properly processed by `du` is limited by the number of files which a single process may have open at once.

2.14 Terminal Characteristics

P. The implementation shall document which terminal types it supports and which of these features and utilities are not supported by each terminal. This implementation-defined list of terminals

-Shall include at least one terminal type that is capable of supporting all of the standard utilities and all of their features, if the {POSIX2_CHAR_TERM} option is provided.

-May group terminals in terms of families or equivalences to other documented terminal types.

-Need not consist of an exhaustive list of terminal models when the implementor considers that some terminal types are used too infrequently to be listed.

S. Any terminal for which a user can obtain a TERMINFO description and run through the `tic` utility is capable of supporting all POSIX.2 utilities.

POSIX.2 Section 3, Shell Command Language

3.5.3 Variables

PS1
P. For users who have additional *implementation-defined* privileges, the default may be another, *implementation-defined* value.

S. If a user's effective ID has the value of zero (0), the default value of **PS1** changes from "$ " to "# ".

3.7 Redirection

P. In this standard, open files are represented by decimal numbers starting with zero. It is *implementation-defined* what the largest value can be; however, all implementations shall support at least 0 through 9 for use by the application.

S. The largest value that can be used as a file descriptor is 9. This means that the user can have up to ten files open, including the standard input, standard output, and standard error files.

POSIX.2 section 4, Execution Environment Utilities

4.1.7.3 Variables and Special Variables

`ENVIRON`
P. In all cases where the behavior of `awk` is affected by environment variables...the environment used shall be the environment at the time `awk` began executing; it is *implementation-defined* whether any modification of `ENVIRON` affects this environment.

S. Changing `ENVIRON` has no effect.

`SUBSEP`
P. The subscript separator string for multidimensional arrays; the default value is *implementation-defined*.

S. The `awk` utility uses the value \034 (the ASCII FS control character).

4.1.7.6.2.3 Input/Output and General Functions

close
P. The limit on the number of open expression arguments is *implementation-defined*.

S. The `awk` utility defines this limit as the value {OPEN_MAX}-4. The minimum value for this limit is 12 and by default, this value is 60.

4.1.7.8 `awk` Lexical Conventions

Table 4-2 \ddd
P. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is *implementation-defined*.

S. The `awk` utility supports 8 bit bytes.

4.2.2 `basename` – Return Nondirectory Portion of Pathname: Description

P. If *string* is `//`, it is *implementation-defined* whether steps (2) through (5) are skipped or processed.

S. "`/`" is returned.

4.5.2 `cd` – Change Working Directory: Description

P. If **HOME** is empty or is undefined, the default behavior is *implementation-defined*.

S. In both cases the `cd` utility gives the following error message:

```
ksh:cd. bad directory
```

4.5.4 Operands

directory
P. If directory is `-`, the results are *implementation-defined*.

S. The `cd` utility handles this case by performing the equivalent of the command `"cd "$OLDPWD" && pwd"`.

4.7.2 `chmod` – Change File Modes: Description

P. It is *implementation-defined* whether and how the `chmod` utility affects any alternate or additional file access control mechanism being used for the specified file.

S. The `chmod` utility does not effect access control lists.

4.7.7 `chmod` – Extended Description

P. When using the symbolic mode form on a regular file, it is *implementation-defined* whether or not:

(1) Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are ignored,

(2) Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-on-execution bits, or

(3) Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear are ignored.

S. (1) `{u|g|a}{+|=}s` are ignored. `{u|g|o|a}{+|=}l` are allowed. The letter `l` is the file-lock bit (equivalent to the S_GID bit).

(2) ID bits are also cleared.

(3) Requests are not ignored. Note: The S_GID bit must be referred to as the letter `l`.

P. When using the symbolic mode form on other file types, it is *implementation-defined* whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bit are honored.

S. In the case of set, the requests are honored when the execution bits are set or being set. In the case of clear, the requests are honored. Note: The S_GID bit must be referred to as the letter `l`.

P. For each bit set in the octal number, the corresponding file permission bit shall be set; all other file permission bits shall be cleared. For regular files, each bit set in the octal number corresponding to the set-user-ID-on-execution or set-group-ID-on-execution bits shall be set; if these bits are not set in the octal

number, they shall be cleared. For other file types, it is *implementation-defined* whether or not request to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

S. The requests are honored.

4.13.2 cp– Copy Files: Description

(2) (c)
P. If *dest_file* exists and it is a file type not specified by POSIX.1, the behavior is *implementation-defined*.

S. The behavior in this case depends upon the underlying behavior of the `stat()` system function when applied with the *dest_file* name. For instance, if *dest_file* was a symbolic link to a directory, `stat()` returns information indicating that *dest_file* was a directory and the behavior will be as if *dest_file* was a directory. For all other non-directory types, the behavior would be the same as (2)(d).

(4) (a)
P. If the `-r` option was specified, the behavior is *implementation-defined*.

S. The `cp` utility defines the behavior of the `-r` option as being similar to the behavior of the `-R` option except when copying SPECIAL files. The `-r` option actually tries to read SPECIAL files, while `-R` recreates them. For example, if the `-R` option is specified and the source file is of type FIFO, then the destination will be another file of type FIFO. If the `-r` option is specified instead, then the destination file will be a regular file consisting of the contents of the FIFO source file.

(4)(b)(2)
P. If *source_file* is a file of type FIFO, the file permission bits shall be the same as those of *source_file*, modified by the file creation mask of the user if the `-p` option was not specified. Otherwise, the permissions, owner ID, and group ID of *dest_file* are *implementation-defined*.

S. The `cp` utility sets the *dest_file* privileges to those of the *source_file* and the owner ID and group ID to the current effective user and group IDs.

P. If the implementation provides additional or alternate access control mechanisms, their effect on copies of files is *implementation-defined*.

S. The `cp` utility defines no alternate or additional file access control mechanisms.

4.13.3 Options

`-p`
P. (3) Other *implementation-defined* bits may be duplicated as well.

S. The `cp` utility duplicates all of the bits of *st_mode* member from the `stat()` function. In particular, on systems supporting the UNIX variable *S_ISVTX*, `cp` duplicates the "sticky bit".

`-r`
P. The treatment of special files is *implementation-defined*.

S. With `-r` the `cp` utility attempts to open the special file, and copy its contents. For example, if the `-R` option is specified, and the source file is of type FIFO, the destination is another file of type FIFO. If, instead, the `-r` option is specified, the destination file will be a regular file, consisting of the contents of the FIFO.

4.18.2 `dirname` – Return Directory Portion of Pathname: Description

P. (6) If the remaining *string* is `//`, it is *implementation-defined* whether steps (7) and (8) are skipped or processed.

S. The `dirname` utility processes steps (7) and (8) when converting a string of `//` to a filename; that is, `dirname //` converts to `/`.

4.19.4 Operands

string
P. If the first operand is "`-n`" or if any of the operands contain a backslash (`\`) character, the results are *implementation-defined*.

S. The `echo` utility takes no special action for "`-n`"; the operand is echoed directly.

The historical SVID functionality as an extension to the standard is supported and includes the following escape sequences in the `echo` operands:

\a	Write an <alert> character.
\b	Write a <backspace> character.
\c	Suppress the <newline> character that otherwise follows the final argument in the output, with everything after \c in input being ignored.
\f	Write a <form-feed> character.
\n	Write a <newline> character.
\r	Write a <carriage-return> character.
\t	Write a <tab> character.
\v	Write a <vertical tab> character.
\\	Write a backslash character.
\0*num*	Write an 8 bit value that is the 0-, 1-, 2-, or 3 digit octal number *num*.
\X	When X is not one of the preceding characters, the echo utility simply echoes the two-character sequence \X.

4.20.7.3.14 List Command

P. If the size of a byte on the system is greater than nine bits, the format used for nonprintable characters is *implementation-defined*.

S. The ed utility does not support byte sizes other than 8 bits.

4.24.4 Operands

exec
P. If a utility_name or argument string contains the two characters {}, but not just the two characters {}, it is *implementation-defined* whether find replaces those two characters with the current pathname or uses the string without change.

S. The find utility uses the string without change.

4.30.2 id– Return User Identity: Description

P. If a user operand is provided and the process has the appropriate privileges, the user and group IDs of the selected user shall be written.

S. In this case, all processes are considered to have "appropriate privileges."

4.33.2 `ln` – Link Files: Description

P. If the last operand specifies an existing file of a type not specified by POSIX.1 {8}, the behavior is *implementation-defined*.

S. If the file type is not specified by POSIX.1, it is treated as a non-directory file.

4.33.4 Operands

source_file
P. A pathname of a file to be linked. This can be a regular or special file; whether a directory can be linked is *implementation-defined*.

S. Links to directories are not supported. Symbolic links, however, to directories are supported.

4.34.3 Options

`-a`
P. The manner in which the implementation determines what other locales are available is *implementation-defined*.

S. The `locale` utility lists each directory in `/usr/lib/locale` when the `-a` option is specified.

4.34.4 Operands

`name`
P. It is *implementation-defined* whether any keyword values are written for the categories **LC_CTYPE** and **LC_COLLATE**.

S. Keyword values are written for **LC_CTYPE**, but not for **LC_COLLATE**.

4.35.2 `localedef` – Define Locale Environment: Description

P. It is *implementation-defined* whether users shall have the capability to create new locales, in addition to those supplied by the implementation. If the symbolic constant {POSIX2_LOCALEDEF} is defined, then the system supports the creation of new locales.

S. Users are permitted to create locales using `localedef`.

4.36.2 `logger` – Log Messages: Description

`-name`
P. It is *implementation-defined* whether messages written in locales other than the POSIX Locale are effective.

S. Messages written in other locales are permitted. The `logger` utility always executes in the POSIX locale and treats the message operand as a sequence of bytes.

4.39.3 Options

`-a`
P. Entries beginning with a period (.) shall not be written out unless they are explicitly referenced, the `-a` option is supplied, or an *implementation-defined* condition causes them to be written.

S. The `ls` utility provides other options, namely `-f` and `-A`, which also enable the listing of entries beginning with a period(.). There are no other conditions that would allow entries beginning with a period (.) to be listed.

4.39.5.3 Environment Variables

COLUMNS
P. If **COLUMNS** is not set or is invalid, an *implementation-defined* number of column positions shall be assumed, based on the knowledge of the output device by the implementation.

S. The `ls` utility obtains the numbers of columns. The approach used to determine the numbers of columns is as follows:

- If the **COLUMNS** environment variable is a valid integer number, then it is used;
- If the **COLUMNS** variable is not a valid integer, use the information returned by the `ioctl(STDOUT_FILENO, TIOCGWINSZ, $wininfo)` system function.
- If the values remain invalid, the value of 80 is used for columns.

4.39.6.1 Standard Output

P. The default format shall be to list one entry per line to standard output. If the output is to a terminal, the format is *implementation-defined*.

S. The ls utility uses a multi-column format, as if the user specified `-C`.

P. If the file is a character special or block special file, the size of the file may be replaced with *implementation-defined* information associated with the device in question.

S. The `ls` utility replaces the file size with the numeric values of the major and minor device numbers displayed with the format "`%u, %u`". The major and minor device numbers are obtained by taking the *st_rdev* structure member returned from the `stat()` system function and passing it to the `major()` and `minor()` system functions. See `stat(2)` and `mkdev(3)` manpages for further explanations.

P. Implementations may add other characters to this list [of *entry type* characters] to represent other, *implementation-defined*, file types.

S. The `ls` utility also uses the character `x` to indicate "none of the above" file types.

The character `s` is used to indicate "sockets".
The character `l` is used to indicate symbolic "links".
The character `L` is used to indicate mandatory locking.

P. Implementations may add other characters to this list for the third character position.

S. The `ls` utility uses the characters `T` or `t` to indicate the "sticky bit" file attribute. This bit is defined as the `S_ISVTX` bit mask defined in `<stat.h>`.

4.40.6.3 Output Files

P. When a message from the system mailbox or entered by the user is not a text file, it is *implementation-defined* how such a message is stored in files written by `mailx`.

S. The mailx utility attempts to process the file and therefore produces inappropriate results.

4.40.7.1 `mailx` Internal Variables

`crt=`*number*
P. Pipe messages having more than *number* lines through the command specified by the value of the **PAGER** variable. The default shall be `nocrt`. If it is set to null, the value used is *implementation-defined.*

S. If it is set to null, the value is whatever the screen size is minus 2.

4.40.7.3 `mailx` Command Escapes

`~h`
P. If standard input is a terminal, prompt for a Subject line and the To, Cc, and Bcc lists. Other *implementation-defined* headers may also be presented for editing.

S. None.

4.43.2 `mv` – Move Files: Description

P. If any operand specifies an existing file of a type not specified by POSIX.1 {8}, the behavior is *implementation-defined*.

S. For symbolic link file types, the behavior of `mv` is to reference the symbolic link file itself when validating the existence of the source file arguments (e.g. uses `lstat()`) and to reference the file to which the symbolic link points when validating and referencing the target argument (e.g. uses `stat()`).

4.45.7 Extended Description

P. The default number of bytes transformed by output type specifiers `d`, `f`, `o`, `u`, and `x` shall correspond to the various C-language types as follows. If the `c89` compiler is present on the system, these specifiers shall correspond to the sizes used by default in that compiler. Otherwise, these sizes are *implementation-defined*. The POSIX.2 standards expands on this with the explanation that "For the type specifier characters `d`, `o`, `u`, and `x`, the default number of bytes shall correspond to the size of the underlying implementation's basic integral data type..."

S. The default number of bytes transformed by the type specifier characters `d`, `o`, `u`, and `x` is 4 bytes (e.g. based on the C-language type `int`).

P. For these specifier characters [`d`, `o`, `u` and `x`], the implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the c-language types `char`, `short`, `int`, and `long`.

S. The optional number of bytes to be converted by the characters C, S, I, and L are 1, 2, 4, and 4 respectively on both SPARC and Intel.

P. For the type specifier character `f`, the default number of bytes shall correspond to the number of bytes in the underlying implementation's basic double precision floating point date type.

S. The default number of bytes used for the type specifier `f` is 8 bytes (e.g. based on the C-language type `double`).

P. The implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the C-language types `float`, `double` and `long double`.

S. The optional number of bytes to be converted for the characters F, D, and L are 4, 8, and 8 respectively on both SPARC and Intel.

P. The byte order used when interpreting numeric values is implementation-defined, but shall correspond to the order in which a constant of the corresponding type is stored in memory on the system.

S. For SPARC, the most significant byte is stored in the lower memory address. For Intel, the least significant byte is stored in the lower memory address.

P. If the size of a byte on the system is greater than nine bits, the format used for nonprintable characters is *implementation-defined*.

S. On Solaris 2.5, the size of a byte is 8 bits.

P. When either the `-j` *skip* or `-N` *count* option is specified along with the `c` type specifier, and this results in an attempt to start or finish in the middle of a multibyte character, the result is *implementation-defined*.

S. When the output starts or finishes in the middle of a multibyte character, the partial multibyte character is displayed as single-byte characters.

4.48.2 `pax` – Portable Archive Interchange: Description

copy
P. If the destination directory is a file of a type not defined by POSIX.1 {8}, the results are *implementation-defined*.

S. Copying will cause an error.

P. The default output archive format shall be *implementation-defined*.

S. *ustar* is the default archive format.

P. The `pax` utility shall determine, in an *implementation-defined* manner, what file to read or write as the next file.

S. `Go` is prompted to continue and `Quit` is prompted to exit while reading the same archive file.

4.48.3 Options

`-a`
P. It is *implementation-defined* which devices on the system support appending.

S. Regular disk files and 4 mm tape drives support appending.

`-p` *string*
P. The string shall consist of the specification characters `a`, `e`, `m`, `o`, and `p`, and/or other *implementation-defined* characters.

S. There are no other file characteristics.

Specification character `e`
P. Preserve the user ID, group ID, file mode bits, access time, modification time, and any other *implementation-defined* file characteristics.

S. There are no other characters.

Specification character `p`
P. Preserve the file mode bits. Other, *implementation-defined* file-mode attributes may be preserved.

S. The implementation also provides a bit identified by S_ISVTX. For a directory, this bit determines whether or not an unprivileged user may delete or rename another user's files from that directory (refer to `chmod(2)` for other files types).

`-x` *format*
P. *Implementation-defined* formats shall specify a default block size as well as any other block sizes supported for character special archive files.

S. There are no *implementation-defined* formats.

4.48.5.2 Input Files

P. The input file named by the archive option-argument, or standard input when the archive is read from there, shall be a file formatted according to one of the specifications in section 10.1 of POSIX.1 {8}, or some other *implementation-defined* format.

S. No other formats are defined.

4.48.6.1 Standard Output

P. In write mode, if `-f` is not specified, the standard output shall be the archive formatted according to one of the specifications in section 10.1 of POSIX.1 {8}, or some other *implementation-defined* format.

S.No other formats are defined.

4.48.6.3 Output Files

P. In write mode, the output file named by the `-f` option argument shall be a file formatted according to one of the specifications in section 10.1 of POSIX.1 {8}, or some other *implementation-defined* format.

S. No other formats are defined.

4.55.7.3 `sed` *Editing Commands*

[2addr]`l`
P. If the size of a byte on the system is greater than nine bits, the format used for non-printable characters is *implementation-defined*.

S. The `sed` utility does not support byte sizes greater than 8 bits.

4.56.5.3 Environment Variables

PS1
P. For users who have specific additional *implementation-defined* privileges, the default may be another *implementation-defined* value.

S. If a user's effective user ID has the value zero (0), the default value of **PS1** changes from "$" to "#".

4.59.2 `stty` – Set the Options for a Terminal: Description

P. The `stty` utility shall set or report on terminal I/O characteristics for the device that is its standard output. Without options or operands specified, it shall report the settings of certain characteristics, usually those that differ from *implementation-defined* defaults.

S. Certain settings are reported when no options or operands have been specified. The following settings are reported when the characteristic is enabled: `markp`, `spacep`, `oddp`, `evenp`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `hupcl`, `clocal`, `loblk`, `line`, `rows`, `columns`, `ypixels`, `xpixels`, `min`, `time`, `ignbrk`, `brkint`, `ignpar`, `parmrk`, `inlcr`, `igncr`, `icrnl`, `iuclc`, `ixoff`, `imaxbel`, `olcuc`, `onlcr`, `ocrnl`, `onocr`, `onlret`, `del-fill`, `nul-fill`, `cr`, `nl`, `tab`, `bs`, `vt`, `ff`, `xcase`, `echonl`, `noflsh`, `tostop`, `echoctl`, `echoprt`, `echoke`, `defecho`, `flusho`, `pendin`, `iexten`.

The following settings are reported when the characteristic is disabled: `-parity`, `-cread`, `-inpck`, `-istrip`, `-ixon`, `-ixany`, `-opost`, `-isig`, `-icanon`, `-echo`, `-echoe`, `-echok`.

In addition:

`intr` is reported if it is set to something other than `^c`,
`quit` is reported if it is set to something other than `^l`,
`erase` is reported if it is set to something other than `^?`,
`kill` is reported if it is set to something other than `^u`,
`eof` is reported if it is set to something other than `^d`,
`eol` is reported if it is defined,
`eol2` is reported if it is defined,
`swtch` is reported if it is set to something other than `^z`,

`start` is reported if it is set to something other than `^q`,
`stop` is reported if it is set to something other than `^s`,
`susp` is reported if it is set to something other than `^z`,
`dsusp` is reported if it is set to something other than `^y`,
`rprnt` is reported if it is set to something other than `^r`,
`flush` is reported if it is set to something other than `^o`,
`werase` is reported if it is set to something other than `^w`,
`lnext` is reported if it is set to something other than `^v`,
`speed` is reported when `ispeed` and `ospeed` are identical, otherwise `ispeed` and `ospeed` are reported.

4.59.4.4 Local Modes

`iexten` (`-iexten`)
P. Enable (disable) any *implementation-defined* special control characters not currently controlled by `icanon`, `isig`, `ixon`, or `ixoff`.

S. Special control characters not controlled by `icanon`, `isig`, `ixon`, or `ixoff` are `veolz`, `vswtch`, `vreprint`, `vdiscard`, `vdsusp`, `vwerase`, `vlnext`.

4.59.4.6 Combination Modes

`sane`
P. Reset all modes to some reasonable, unspecified values.

S. `stty sane` is equivalent to:

```
stty cs7 parenb cread -csize -parodd -clocal brkint ignpar \
istrip icrnl ixon imaxbel -ignbrk -parmrk -inpck -inlcr -igncr \
-iuclc -ixoff -ixany isig icanon iexten echo echok echoe echoke\
echoctl -xcase -echonl -noflsh opost onlcr -olcuc -ocrnl -onocr \
-onlret -ofill -ofdel -nldly -crdly -tabdly -bsdly -vtdly -ffdly \
erase ^? kill ^u quit ^l intr ^c eof ^d eol undef
```

4.62.4 Operands

P. Additional *implementation-defined* operators and *primary-operators* may be provided by implementations. The additional implementation-defined operators "(" and ")" may also be provided by implementations.

S. The `test` utility provides the following additional primaries:

`-a`	True if both *expression1* and *expression2* are true,
`-o`	True if either *expression1* or *expression2* are true,
`(,)`	Allow primaries to be grouped as single expressions, for use with `-a` and `-o` primaries.
`-k`	True if the "sticky bit" is on
`-nt`	True if *file1* is newer than *file2*
`-ot`	True if *file1* is older than *file2*
`-ef`	True if *file1* has the same device and inode as *file2*; that is, same file
`-L`	True if *file* is a symbolic link
`-h`	True if *file* is a symbolic link

4.63.3 Options

P. The range of valid times past the Epoch is *implementation-defined*...

S. The range of valid times past the Epoch depends on the size of the ANSI C arithmetic type `time_t` and the behavior of the ANSI C `mktime()` routine. `time_t` is a signed long, 32 bits. The maximum positive integer is 2147483647, which represents 68 years, 18 days, 3 hours, 14 minutes, and 7 seconds.

From the Epoch (12:00:00 a.m. January 1, 1970), 2147483647 represents the date: Tue Jan 19 03:14:07 GMT 2038.

For the behavior of `mktime()` wee the `mktime(3C)` man page.

4.64.7 Extended Description

\octal
P. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is *implementation-defined*.

S. The `tr` utility does not support byte sizes greater than 8 bits.

4.68.2 `uname` – Return System Name: Description

P. When options are specified, symbols representing one or more system characteristics shall be written to the standard output. The format and contents of the symbols are *implementation-defined*.

S. The `uname` utility supports seven symbols, including the five specified by POSIX.1, section 4.4.1.2 and displays the additional symbols at the end of the output line.

4.68.6.1 Standard Output

P. Additional *implementation-defined* symbols may be written.

S. The additional symbols are machine class and machine processor type. "machine" symbol is called "platform" in `uname` man page.

POSIX.2 Section 5, User Portability Utilities Option

5.2.2 Execute Commands at a Later Time: Description

P. The at-job shall be executed in a separate invocation of the shell, running in a separate process group with no controlling terminal, except that the environment variables, current working directory, file creation mask, and other *implementation-defined* execution-time attributes in effect when the `at` utility is executed shall be retained and used when the at-job is executed.

S. The limits controlled by the `unlimit` facility are also retained, but only applies to `ksh`, `sh`, and `/usr/xpg4/bin/sh`.

5.2.3 Options

`-m`
P. If `-m` is not used, the standard output and standard error of the job shall be provided to the user via an *implementation-defined* mechanism, unless they are redirected elsewhere;

S. stdout and stderr are sent via `/bin/mail`.

`-q` *queuename*
P. By default, at-jobs shall be scheduled in queue "a". In contrast, queue "b" shall be reserved for batch jobs. The meanings of all other *queuenames* are *implementation-defined*.

S. The `at` utility can use any single-byte character except NULL,'\b', '\n', '\t', and '#' characters as a queuename. All queues have the same functionality as queue 'a', except queue limits and priorities may be set in the `queuedefs` file. See POSIX.2 `queuedefs(4)man` page for more information. 'a' through 'z' are valid queuenames.

5.2.4 Operands

timespec: time
P. The acceptable time-zone names are *implementation-defined*.

S. ZULU, UTC, and GMT (case insensitive)

`%token timezone_name`
P. The name of an optional time-zone suffix to the time field, in an *implementation-defined* manner.

S. ZULU, UTC, and GMT (case insensitive)

5.5.2 `crontab` – Schedule Periodic Background Work: Description

P. If standard output and standard error are not redirected by commands executed from the crontab entry, any generated output or errors shall be mailed, via an *implementation-defined* method, to the user.

S. stdout and stderr are sent via `/bin/mail`.

5.7.4 Extended Description

P. The handling of other files is *implementation-defined*.

S. The `ctags` utility handles C, C++, Pascal, FORTRAN, YACC, and LEX sources.

5.7.7 Operands

P. It is *implementation-defined* what other objects (including duplicate identifiers) produce output.

S. Duplicate identifiers and too many entries produce output.

5.8.6.1 Standard Output

`-P` **<***total space***>**
P. The total size of the file system in 512 B units. The exact meaning of this figure is *implementation-defined*, but should include **<***space used***>**, **<***space free***>**, and any space reserved by the system not normally available to a user.

S. The value of **<***total space***>** comes from the *f_blocks* member of the struct `statvfs(2)` function. The `df` utility uses this *f_blocks* member to represent the total number of data blocks in the file system.

`-P` **<** *space free***>**
P. When this figure is less than or equal to zero, it shall not be possible to create any new files on the file system without first deleting others, unless the process has appropriate privileges.

S. "Appropriate privileges" means processes with the effective id of '0' (e.g. root authority).

5.10.7.2.5 `chdir`

P. If **HOME** is empty or is undefined, the default behavior is *implementation-defined*.

S. The `ex` utility prints the following error: No such file or directory.

5.10.7.2.13 `list`

P. Write the addressed lines in a way that should be unambiguous: nonprintable characters shall be written as *implementation-defined* multicharacter sequences...

S. The `ex` utility displays nonprintable characters in the current locale as a circumflex (^) followed by a single character or two uppercase hexadecimal digits. If the nonprintable character is a control character defined in Table 2-20 of the POSIX.2 standard, the corresponding alphabetic character in the table is displayed after the circumflex. Otherwise, the two uppercase hexadecimal digits displayed are those that represent the actual value of the nonprintable character.

5.10.7.2.14 `map`

P. Implementations may restrict the set of commands accepted within *rhs*; the list of restrictions is *implementation-defined*.

S. There are no restrictions on *rhs*.

5.10.7.2.21 `print`

P. Nonprintable characters, except for <tab>, shall be written as *implementation-defined* multicharacter sequences.

S. Nonprintable characters are written in octal.

5.10.7.2.29 `source`

P. The maximum supported nesting depth is *implementation-defined*, but shall be at least one.

S. The `ex` utility nesting is limited by the number of `dup(2)` system calls that a single process can have. `dup(2)` is limited via `getrlimit(2)`.

5.10.7.2.37 `write`

P. If *file* is specified and is not the current file, and the file named by *file* exists, then the write shall fail. If the current file has been changed by the `file` command, and that files exist, the write shall fail. In either case, the write can be forced by appending the character `!`to the command name. An existing file can be appended to by appending >> to the command name. If the file does not exist, the result is *implementation-defined*.

S. The write succeeds if one has write permission in the directory.

5.10.7.5.8 `list`

P. If `list` is set, write the addressed lines in a way that should be unambiguous: non printable characters shall be written as *implementation-defined* multicharacter sequences; the end of the line shall be marked with a `$`.

S. Nonprintable characters are written in octal.

5.10.7.5.12 `paragraphs, para`

P. The `sections` option can be set to a character string consisting of zero or more character pairs. The default value is *implementation-defined*.

S. The default values are JP, LP, PP, OP, P , LI, pp, lp, ip, np, and bp.

5.10.7.5.18 `sections`

P. The default value is *implementation-defined*.

S. The default values are NH, SH, H , HU, uh, sh, and +c.

5.10.7.5.24 `tags`

P. By default, filenames of `tags` shall be searched for in the current directory and in other *implementation-defined* directories.

S. Filenames of `tags` are also searched for in `/usr/lib/tags`.

5.10.7.5.28 `window`

P. The baud rate of the terminal line may reduce the default in an *implementation-defined* manner.

S. For baud rate less than 1200, terminal lines equal 8. For baud rates greater than or equal to 1200 but less than 2400, terminal lines equal 16.

5.12.2 `fc` – Process Command History List: Description

P. When the number reaches an *implementation-defined* upper limit, which shall be no smaller than the value in **HISTSIZE** or 32767 (whichever is greater), the shell may wrap the numbers, starting the next command with a lower number (usually 1).

S. The upper limit is the maximum positive integer or 2147483647.

5.12.5.3 Environment Variables

HISTFILE
P. An implementation may choose to access this variable only when initializing the history file; this initialization shall occur when `fc` or `sh` first attempts to retrieve entries from, or add entries to, the file as the result of commands issued by the user, the file named by the **ENV** variable, or implementation-defined startup files. Therefore, it is *implementation-defined* whether changes made to **HISTFILE** after the history file has been initialized are effective.

S. The **HISTFILE** environment variable is examined when the history file is opened for the first time. If the **HISTFILE** environment variable is changed, then the current history file is closed and a new history file is opened according to the new value of the **HISTFILE** environment variable.

P. Implementations may choose to disable the history list mechanism for users with appropriate privileges who do not set **HISTFILE**; the specific circumstances under which this will occur are *implementation-defined*.

S. There are no circumstances under which the history list mechanisms are disabled.

P. An implementation may choose to access this variable only when initializing the history file, as described under **HISTFILE**. Therefore, it is *implementation-defined* whether changes made to **HISTFILE** after the history file has been initialized are effective.

S. The **HISTFILE** environment variable is examined when the history file is opened for the first time. If the **HISTFILE** environment variable is changed, then the current history file is closed and a new history file is opened according to the new value of the **HISTFILE** environment variable.

5.14.2 `file` – Determine File Type: Description

P. If the file is not a regular file, its file type shall be identified. The file types directory, FIFO, block special, and character special shall be identified as such. Other *implementation-defined* file types may also be identified.

S. The `file` utility uses the information contained in the file named `/etc/magic` to identify the file types stated in the POSIX.2 standard.

5.16.2 `man` – Display System Documentation: Description

P. If more information is available, the `man` utility shall provide it in an *implementation-defined* manner.

S. All the information that `man` provides is displayed in the same manner; as a sequence of text characters obtained from a file found in the directories specified by the user's MANPATH environment variable.

5.16.6.1 Standard Output

P. The `man` utility writes text describing the syntax of the utility *name*, its options, and it operands or, when `-k` is specified, lines from the summary database. The format of this text is *implementation-defined*.

S. The `man` utility displays the contents of a file (as described in section 5.16.2 above) after being processed by the command "`nroff -mansun`".

5.17.2 `mesg` – Permit or Deny Messages: Description

P. Processes with appropriate privileges may be able to send messages to the terminal independent of the current state.

S. *The POSIX 1003.2 Shell and Utility Application Interface* does not allow messages to be sent to a terminal if its state does not permit it.

5.18.3 Options

`-u`
P. Treat **<**`backspace`**>** as a printable control character, displayed as an *implementation-defined* character sequence suppressing backspacing and the special handling that produces underlined or standout-mode text on some terminal types.

S. The `more` utility displays the `<backspace>` as the two character string `^H`.

5.18.7 Extended Description

P. It is *implementation-defined* how other nonprintable characters are written.

S. Other nonprintable characters are written as Ctrl-*letter* and ESC-*letter*.

P. In the case that text is being taken from a nonrewinding stream, such as a pipe, it is *implementation-defined* how much backwards motion is supported.

S. The `more` utility sets no limit to the amount of backward motion supported, other than the amount of free space in the `/tmp` directory (or the **TMPDIR** directory, if defined.)

5.18.7.1 Help

P. Write a summary of these command and other *implementation-defined* commands.

S. The `more` utility implements an additional command called `!` which is included in this summary.

5.18.7.24 Invoke Editor

P. It is *implementation-defined* whether line-setting options are passed to editors other than `vi` and `ex`.

S. The `more` utility does not pass line-setting option to any other editors.

P. The file types that can be edited are *implementation-defined*.

S. Only files which are acceptable to the editor being invoked may be edited successfully.

5.19.2 `newgrp` – Change to a New Group: Description

P. If no password is required for the specified group, it is *implementation-defined* whether users not listed as members of that group can change to that group. Whether or not a password is required, *implementation-defined* system accounting or security mechanisms may impose additional authorization restrictions.

S. Access to the group is denied if a group has no password assigned to it and the user is not a member of the group.

5.20.2 `nice` – Invoke a Utility with an Altered System Scheduling Priority: Description

P. With no options and only if the user has appropriate privileges, the executed utility shall be run with a system scheduling priority that is some *implementation-defined* quantity less than or equal to the system scheduling priority of the current process.

S. With no options, the `nice` utility increments the system scheduling priority by a value of 10.

P. If the user lacks appropriate privileges to affect the system scheduling priority in the requested manner, the `nice` utility shall not affect the system scheduling priority; in this case, a warning message may be written to the standard error, but this shall not prevent the invocation of *utility* or affect the exit status.

S. The user's ability to alter the system scheduling priority depends on whether or not the process is a "time-sharing" (TS) process and depends on the appropriate privileges required by the `priocntlset(2)` and `priocntl(2)` functions. The `nice` utility only operates on processes that are in the "time-sharing" class and uses the `priocntl(2)` function to get the process current nice value, and then uses `priocntlset()` to increment/decrement this nice value.

5.20.3 Options

`-n` *increment*
P. If the *increment* option argument shall be a positive or negative decimal integer that shall be used to modify the system scheduling priority of the executed utility in an implementation-defined manner.

S. SunOS has a notion of "nice"-ness and this value shall increment or decrement the processes "nice" value. This "nice" value is used to determine the system scheduling priority. This is done using the `priocntlset(2)` function.

5.21.5.2 Input Files

P. The `nm` utility may accept additional *implementation-defined* object library formats for the input file.

S. The `nm` utility does not accept any additional object library formats.

5.21.6.1 Standard Output

P. Symbol type, which shall either be one of the following single characters or an *implementation-defined* type represented by a single character:

`A`	Global absolute symbol
`a`	Local absolute symbol
`B`	Global bss symbol
`b`	Local bss symbol
`D`	Global data symbol
`d`	Local data symbol
`T`	Global text symbol
`t`	Local text symbol
`U`	Undefined symbol

S. The `nm` utility uses the following types:

`B`	Global bss symbol
`b`	Local bss symbol
`D`	Global data symbol
`d`	Local data symbol
`T`	Global text symbol
`t`	Local text symbol
`U`	Undefined symbol
`n`	Locally no defined type
`N`	Globally no defined type

5.23.2 `ps` – Report Process Status: Description

P. When the `-o` option is not specified, information about processes selected shall be written in an *implementation-defined* manner.

S. When the `-f` option is used, process information shall be displayed as if the following was specified on the command line:

-o pid,ppid, c
-o stime,tty=TTY
-o time=TIME
-o args=CMD

When the `-j` option is used, process information shall be displayed as if the following was specified on the command line:

-o pid,ppid,sid
-o time=TIME
-o fname=CMD

When the `-l` option is used, process information shall be displayed as if the following was specified on the command line:

-o pid,tty=TTY
-o time=TIME
-o fname=CMD

5.23.3 Options

`-t` *termlist*
P. Terminal identifiers shall be given in an *implementation-defined* format.

S. Terminal identifiers shall be accepted and displayed in the same format as is stored in the system `/etc/utmp` file which is the same format as is displayed by the `who` utility.

5.23.61 Standard Output

P. When the `-o` option is not specified, the standard output is *implementation-defined*.

S. Without options, `ps` prints information about processes associated with the controlling terminal. The output contains only the process ID, terminal identifier, cumulative execution time, and the command name.

`args`
P. The implementation may truncate this value to the field width; it is *implementation-defined* whether any further truncation occurs.

S. The maximum number of characters displayed is limited to 80.

P. Any *implementation-defined* variables shall specify in the conformance document if the field may contain <`blank`>s as well as the default header.

S. No implementation-defined variables contain <`blank`>s.

Table 8-1 lists the Solaris implementation format specifiers and the default header used with each.

Table 9-1 Solaris Implementation Format Specifiers and their Default Header

Format Specifier	Default Header	Format Specifier	Default Header
addr	ADDR	pri	PRI
c	C	rgid	RGID
class	CLS	rss	RSS
f	F	ruid	RUID
fname	COMMAND	s	S
gid	GID	sid	SID
opri	PRI	stime	STIME
osz	SZ	uid	UID
pmem	%MEM	whan	WCHAN

5.24.2 `renice` – Set System Scheduling Priorities of Running Processes: Description

P. The system scheduling priority shall be bounded in an *implementation-defined* manner. If the requested increment...would raise or lower the system scheduling priority of the executed utility beyond *implementation-defined* limits, then the limit whose value was exceeded shall be used.

S. See section 5.20.3 *Options* {*of* `nice`} in this document.

P. Regardless of which options are supplied or any other factor, `renice` shall not alter the system scheduling priorities of any process unless the user requesting such a change has appropriate privileges to do so for the specified process. If the user lacks appropriate privileges to perform the requested action, the utility shall return an error status.

S. See section 5.20.2 *Description* {*of* `nice`} in this document.

5.24.3 Options

`-n` *increment*
P. Negative *increment* values may require appropriate privileges and shall cause a higher system scheduling priority.

S. See section 5.20.3 *Options* {*of* `nice`} in this document.

5.26.2 `strings` – Find Printable Strings in Files: Description

P. Additional *implementation-defined* strings may be written.

S. No other strings are written.

5.26.3 Options

`-a`
P. If `-a` is not specified, it is *implementation-defined* what portion of each file is scanned for strings.

S. The `strings` utility ignores the portion of an executable file containing executable instructions.

5.28.2 `talk` – Talk to Another User: Description

P. When and only when the `stty` *iexten* local mode is enabled, additional special control characters and multibyte or single-byte functions shall be processed in an *implementation-defined* manner.

S. The `talk` utility does not support additional special control characters and multibyte or single-byte functions.

P. Typing other nonprintable characters shall cause *implementation-defined* sequences of printable characters to be sent to the terminal of the recipient.

S. Other nonprintable characters are displayed on recipient's terminal as "?" character.

P. However a user's privilege may further constrain the domain of accessibility of other user's terminals. The `talk` utility shall fail when the user lacks the appropriate privileges to perform the requested action.

S. The `talk` utility does not further constrain the domain of accessibility of other users terminals beyond that imposed by the `mesg` utility.

5.30.4 Operands

`init`
P. Display the sequence that will initialize the terminal of the user in an *implementation-defined* manner.

S. The `tput` utility, the TERMINFO database defines this sequence for each terminal.

`reset`
P. Display the sequence that will reset the terminal of the user in an *implementation-defined* manner.

S. The `tput` utility, the TERMINFO database defines this sequence for each terminal.

5.36.2 `who` – Display Who is on the System: Description

P. The `who` utility shall list various pieces of info about accessible users. The domain of accessibility is *implementation-defined*.

S. The `who` utility shows information for all users.

5.36.6.1 Standard Output

P. The `who` utility writes its default information to the standard output in an *implementation-defined* format, subject only to the requirement of containing the information [in the standard].

S. The `who` utility writes its default information to the standard output in the following format:

`%s %s %s %s`, <*user name*>,<*terminal name*>,<*time of login*>

5.37.2 `write` – Write to Another User: Description

P. When and only when the `stty` *iexten* local mode is enabled, additional special control characters and multibyte or single-byte functions shall be processed in an *implementation-defined* manner.

S. The `write` utility does not support any additional special control characters or multiple or single-byte functions.

P. Typing other nonprintable characters shall cause *implementation-defined* sequences of printable characters to be written to the terminal of the recipient.

S. Other nonprintable characters are displayed on recipient's terminal as "?" character.

P. To write to a user who is logged in more than once, the terminal argument can be used to indicate which terminal to write to; otherwise, the recipient's terminal is selected in an *implementation-defined* manner and an informational message shall be written to the sender's standard output, indicating which terminal was chosen.

S. The terminal selected is taken from the first entry in the `/etc/utmp` file that contains the user id that matches the recipients user id.

P. However, a privilege of a user may further constrain the domain of accessibility of the terminals of other users. The `write` utility shall fail when the user lacks the appropriate privileges to perform the requested action.

S. The `write` utility does not further constrain the domain of accessibility of other users terminals beyond that imposed by the `mesg` utility.

POSIX.2 Section 6, Software Development Utilities Option

6.2.7.1 Makefile Syntax

P. If `./makefile` is not found, the file `./Makefile` shall be tried. If neither `./makefile` nor `./Makefile` are found, other *implementation-defined* pathnames may also be tried.

S. The following pathnames are tried in the order given:

./makefile
./Makefile
s.makefile
s.Makefile
SCCS/s.makefile
SCCS/s.Makefile

6.2.7.2 Makefile Execution

P. The macros from the command line to `make` shall be added to the environment of `make`. Other *implementation-defined* variables may also be added to the environment of `make`.

S. The **MAKEFLAGS** variable is added if it did not already exist.

P. If the **MAKEFLAGS** variable is not set in the environment in which `make` was invoked, in the makefile, or on the command line, it shall be created by `make` and shall contain all options specified on the command line except for the `-f` and `-p` options. It may also contain *implementation-defined* options.

S. The make utility, **MADEFLAGS** may also contain the `-E`, `-V`, `-v`, and `-x` options, if they appeared on the original make command line. See the `man` page for `make`, in the SUNWp2man package, for a description of these options.

6.2.7.3 Target Rules

P. The interpretation of targets containing the characters "`%`" and " `"` " is *implementation-defined.*

S. The `make` utility treats targets containing "%" as meta-rules unless the user specifies the `.POSIX` special target, in which case, it ignores meta-rules. The make utility uses the " `"` " character in pairs for quoting. It treats a # character contained within a " `"` " pair as though it has no special meaning.

6.2.7.4 Macros

P. If `SHELL` is defined in the makefile or is specified on the command line, it shall replace the original value of the `SHELL` macro, but shall not affect the **SHELL** environment variable. Other effects of defining `SHELL` in the makefile or on the command line are *implementation-defined.*

S. If the user specifies the `.POSIX` special target, the effect of defining `SHELL` is as specified in the POSIX.2 standard. If the user does not specify `.POSIX` and defines `SHELL` as a macro in the makefile, `make` uses the shell specified by `SHELL` but does not change the value of the **SHELL** environment variable in the environment passed to child processes unless the user specified the `-x` option. If the user does not specify `.POSIX` and includes `SHELL=`*shell_path* on the command line, `make` uses *shell_path* as its shell and also assigns it as the value of the **SHELL** environment variable in the environment passed to child processes.

POSIX.2 Annex A: C Language Development Utilities Option

A.1.3 Options

`-D`
P. Additional *implementation-defined names* may be provided by the compiler.

S. The following additional names are provided by the C compiler:

unix
sparc (SPARC only)
i386 (x86 only)
sun

The above are not pre-defined in -Xc mode.

The following predefinitions are valid in all modes:

_sparc (SPARC only)
_i386 (x86 only)
_unix
_sun
_BUILTIN_VA_ARC_INCR
_SUNPRO_C=0x301
_SVR4

A.1.4 Operands

file`.a`
P. Implementations may recognize *implementation-defined* suffixes other than `.a` as denoting object file libraries.

S. None

file`.o`
P. Implementations may recognize *implementation-defined* suffixes other than `.o` as denoting object file libraries.

S. None

`-L` *library*
P. Implementation may recognize *implementation-defined* suffixes other than `.a` as denoting libraries.

S. None

A.1.5.2 Input Files

P. Additional input file formats are *implementation-defined.*

S. The C compiler recognizes the additional input file format: `.il` files (inline template files.)

A.1.7.2 External Symbols

P. The C compiler and link editor shall support the significance of external symbols up to a length of at least 31 bytes; the action taken upon encountering symbols exceeding the *implementation-defined* maximum symbol length is unspecified.

S. The C compiler supports up to 1023 characters.

P. The compiler and link editor shall support a minimum of 511 external symbols per source or object file, and a minimum of 4095 external symbols total. A diagnostic message shall be written to the standard output if the *implementation-defined* limit is exceeded; other actions are unspecified.

S. The number of symbols allowed by the C compiler per source file is dynamic and only limited to the memory and disk space available on the system.

A.2.6.1 Standard Output

P. If the `-t` option is not specified: (1) *implementation-defined* informational, error, and warning messages concerning the contents of `lex` source code input shall be written to either the standard output or standard error.

S. The `lex` utility writes all information, error and warning messages to the standard error. For details of the text of those messages, see section A.2.6.2 Standard Error.

P. If the `-v` option is specified and the `-n` option is not specified, `lex` statistics also shall be written to either the standard output or standard error, in an *implementation-defined* format.

S. The `lex` utility writes these statistics to the standard error. For details of the format of those statistics, see section A.2.6.2 Standard Error.

A.2.6.2 Standard Error

P. If the `-t` option is specified, *implementation-defined* informational, error, and warning messages concerning the contents of `lex` source code input shall be written to standard error.

S. The lex utility writes all information, error and warning messages to the standard error. The following list shows these messages:

```
"Error: EOF in string or character constant"
"Error: EOF inside comment"
"Error: Non-terminated string or character constant"
"Error: Unexpected EOF inside comment"
"Error: Action does not terminate"
"Error: Can't open %s"
"Error: Cannot open file -- %s"
"Error: Cannot read from -- %s"
"Error: Character %o used twice"
"Error: Character range specified between different codesets."
"Error: Character value %d out of range"
"Error: Definitions too long"
"Error: EOF before %%%%"
"Error: EOF inside comment"
"Error: Illegal definition"
"Error: Illegal rule"
"Error: Invalid request %s"
"Error: Non-ASCII characters in start condition."
```

```
"Error: None-ASCII characters in start condition."
"Error: Parse error"
"Error: Parse tree too big %s"
"Error: Premature EOF"
"Error: Start conditions too long"
"Error: Too complex rules -- requires too many char groups."
"Error: Too late for %array"
"Error: Too late for %pointer"
"Error: Too late for language specifier"
"Error: Too many (>%d) pattern-action rules."
"Error: Too many characters pushed"
"Error: Too many definitions"
"Error: Too many large character classes"
"Error: Too many packed character classes"
"Error: Too many positions %s"
"Error: Too many start conditions used"
"Error: Too many start conditions"
"Error: Too many states %s"
"Error: Too many transitions %s"
"Error: \Character table (%t) is supported only in ASCII
    compatibility mode.\n"
"Error: bad translation format
"Error: can't have negative iteration"
"Error: ch table needs redeclaration"
"Error: definition %ws not found"
"Error: definition too long"
"Error: definitions too long"
"Error: executable statements should occur right after %%%%"
"Error: illegal extra \"}\""
"Error: illegal extra slash"
"Error: illegal number of packed character class"
"Error: illegal number of parse tree nodes"
"Error: illegal operator -- %c"
"Error: illegal position number"
"Error: illegal size of output array"
"Error: illegal state number"
"Error: illegal translation number"
"Error: incomplete translation format"
"Error: iteration range must be positive"
"Error: missing translation value"
"Error: output table overflow"
"Error: string name too long"
"Error: unacceptable statement"
"Error: undefined action string"
"Error: undefined start condition %ws"
"Error: unexpected EOF before %%%%"
```

```
"Error: unmatched hyphen"
"Warning: Character '%wc' used twice"
"Warning: No translation given - null string assumed"
"Warning: Non-portable Character Class"
"Warning: Non-terminated string"
"Warning: String too long"
"Warning: Substitution strings may not begin with digits"
"Warning: \"%c\" redefined inside brackets"
"Warning: \\a is ANSI C \"alert\" character"
"Warning: \\x is ANSI C hex escape"
"Warning: bad state %d %o"
"Warning: bad transition %d %d"
"Warning: invalid string following %%%% be ignored"
"Warning: string too long"
"Warning: the values between braces are reversed"
"Warning: undefined string"
```

P. If the `-t` option is not specified:

`(1)` *implementation-defined* informational, error, and warning messages concerning the contents of `lex` source code input shall be written to either the standard output or standard error.

S. The `lex` utility writes all information, error and warning messages to the standard error. See section A.2.6.2 for a list of these messages.

P. (2) If the `-v` option is specified and the `-n` option is not specified, `lex` statistics also shall be written to either the standard output or standard error, in an *implementation-defined* format.

S. The `lex` utility writes these statistics to the standard error. The following list shows these errors:

```
"%d/%d nodes(%%e), %d/%d positions(%%p), %d/%d (%%n),
    %ld transitions, \n"
"%d/%d packed char classes(%%k),"
"%d/%d packed transitions(%%a),"
%d/%d output slots(%%o) \n"
```

A.2.7 Extended Description

P. The input string that was matched is left in *yytext* as a null-terminated string; *yytext* is either an external character array or a pointer to a character string. As explained in A.2.7.1, the type can be explicitly selected using the `%array` or `%pointer` declarations, but the default is *implementation-defined*.

S. By default, the `lex` utility behaves as if the user had explicitly selected the %array declaration.

A.2.7.1 `lex` Definitions

P. The default type of `yytest` is *implementation-defined*.

S. The default type of yytext is `char[]`.

P. In [Table A-1], *n* represents a positive decimal integer, preceded by one or more <`blank`>s. The exact meaning of these table size numbers is *implementation-defined*. The implementation shall document how these numbers affect the `lex` utility and how they are related to any output that may be generated by the implementation should space limitations be encountered during the execution of `lex`.

S. The following table size declarations represent settable limits: `%p`, `%n`, `%a`, `%e`, `%k`, and `%o`. The lex statistics can be used to show the table sizes, which are set by default using the values shown in table

Table 9-2

Declaration	Description	Default
%p n	number of positions	2500
%n n	number of states	500
%a n	number of transitions	2000
%e n	number of parse tree nodes	1000
%k n	number of packed character classes	10000
%o n	size of the output array	3000

Depending on the system configuration and the available resources, limits may affect what input `lex` can successfully compile.

A.2.7.4 `lex` Regular Expressions

Table A-2 *digits*
P. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is *implementation-defined*.

S. The `lex` utility does not support byte sizes greater than 8 bits.

A.3.6.3.3 `yacc` Description File

P. Limits for internal tables also shall be reported in an *implementation-defined* manner.

S. The `yacc` expands the internal tables as needed. The only limitation is system memory.

A.3.7.9 Limits

P. The exact meaning of these values is *implementation-defined*. The *implementation shall define* the relationship between these values and between them and any error messages that the implementation may generate should it run out of space for any internal structure.

S. The internal tables are allocated dynamically. When system memory gets short and allocation of memory fails, one of the following error messages is issued:

"Couldn't allocate initial table"
"Could not allocate lookset array"
"Cannot allocate tables in mktbls"

9