-- --

IEEE P1003.2 Draft 11.2 – September 1991

4.56.6.3 Output Files

None.

4.56.7 Extended Description

See Section 3.

4.56.8 Exit Status

The `sh` utility shall exit with one of the following values: 1

- 0 The script to be executed consisted solely of zero or more blank lines 1
  and/or comments. 1
- 1−125 A noninteractive shell detected a syntax, redirection, or variable 1
  assignment error. 1
- 127 A specified *command_file* could not be found by a noninteractive 1
  shell. 1

Otherwise, the shell shall return the exit status of the last command it invoked or attempted to invoke (see also the `exit` utility in 3.14.7).

4.56.9 Consequences of Errors

See 3.8.1.

4.56.10 Rationale. *(This subclause is not a part of P1003.2)*

Examples, Usage

```
sh -c "cat myfile"
```

```
sh my_shell_cmds
```

The `sh` utility and the `set` special built-in utility share a common set of options. Unlike `set`, however, the POSIX.2 `sh` does not specify the use of `+` as an option flag, because it is not particularly useful (the `+` variety generally invokes the default behavior) and because *getopt*() does not support it. However, since many historical implementations do support the plus, applications will have to guard against the relatively obscure case of a first operand with a leading plus sign.

There is a large number of environment variables used by historical implementations of `sh` that will not be introduced by POSIX.2 until the UPE is completed.

The KornShell ignores the contents of **IFS** upon entry to the script. A conforming application cannot rely on importing **IFS**. One justification for this, beyond security considerations, is to assist possible future shell compilers. Allowing **IFS** to be imported from the environment will prevent many optimizations that might otherwise be performed via dataflow analysis of the script itself.

The standard input and standard error are the files that determine whether a shell is interactive when `−i` is not specified. For example,

`sh > file` and `sh 2> file`

create interactive and noninteractive shells, respectively. Although both accept terminal input, the results of error conditions will be different, as described in 3.8.1; in the second example a redirection error encountered by a special built-in utility will abort the shell.

The text in Standard Input about nonblocking reads concerns an instance of `sh` 1
that has been invoked, probably by a C-language program, with standard input 1
that has been opened using the O_NONBLOCK flag; see POSIX.1 {8} *open*(). If the 1
shell did not reset this flag, it would immediately terminate because no input data 1
would be available yet and that would be considered the same as end-of-file. 1

History of Decisions Made

See the Rationale for Section 3 concerning the lack of interactive features in `sh`. These features, including optional job control, are scheduled to be added in the User Portability Extension.

The **PS1** and **PS2** variables are not specified because this standard, without UPE, does not describe an interactive shell.

The options associated with a *restricted shell* (command name `rsh` and the `−r` option) were excluded because the developers of the standard felt that the implied level of security was not achievable and they did not want to raise false expectations.

On systems that support set-user-ID scripts, a historical trapdoor has been to link a script to the name `−i`. When it is called by a sequence such as `sh -` or by `#! /bin/sh -` the historical systems have assumed that no option letters follow. Thus, POSIX.2 allows the single hyphen to mark the end of the options, in addition to the use of the regular `--` argument, because it was felt that the older practice was so pervasive. An alternative approach is taken by the KornShell, where real and effective user/group IDs must match for an interactive shell; this behavior is specifically allowed by POSIX.2. (Note: there are other problems with set-user-ID scripts that the two approaches described here do not deal with.)

4.57 `sleep` — Suspend execution for an interval

4.57.1 Synopsis

`sleep` *time*

4.57.2 Description

The `sleep` utility shall suspend execution for at least the integral number of seconds specified by the *time* operand.

4.57.3 Options

None.

4.57.4 Operands

The following operands shall be supported by the implementation:

time A nonnegative decimal integer specifying the number of seconds for which to suspend execution.

4.57.5 External Influences

4.57.5.1 Standard Input

None.

4.57.5.2 Input Files

None.

4.57.5.3 Environment Variables

The following environment variables shall affect the execution of `sleep`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

LC_MESSAGES This variable shall determine the language in which messages should be written.

4.57.5.4 Asynchronous Events

If the `sleep` utility receives a SIGALRM signal, one of the following actions shall be taken:

(1) Terminate normally with a zero exit status

(2) Effectively ignore the signal

(3) Provide the default behavior for signals described in 2.11.5.4. This could include terminating with a nonzero exit status.

The `sleep` utility shall take the standard action for all other signals; see 2.11.5.4.

4.57.6 External Effects

4.57.6.1 Standard Output

None.

4.57.6.2 Standard Error

Used only for diagnostic messages.

4.57.6.3 Output Files

None.

4.57.7 Extended Description

None.

4.57.8 Exit Status

The `sleep` utility shall exit with one of the following values:

0 The execution was successfully suspended for at least *time* seconds, or a SIGALRM signal was received (see 4.57.5.4).

>0 An error occurred.

4.57.9 Consequences of Errors

Default.

4.57.10 Rationale. *(This subclause is not a part of P1003.2)*

Examples, Usage

The exit status is allowed to be zero when `sleep` is interrupted by the SIGALRM signal, because most implementations of this utility rely on the arrival of that signal to notify them that the requested finishing time has been successfully attained. Such implementations thus do not distinguish this situation from the successful completion case. Other implementations are allowed to catch the signal and go back to sleep until the requested time expires or provide the normal signal termination procedures.

History of Decisions Made

As with all other utilities that take integral operands and do not specify subranges of allowed values, `sleep` is required by this standard to deal with *time* requests of up to 2 147 483 647 seconds. This may mean that some implementations will have to make multiple calls to the underlying operating system's delay mechanism if its argument range is less than this.

4.58 `sort` — Sort, merge, or sequence check text files

4.58.1 Synopsis

`sort [–m] [–o` *output* `] [–bdfinru] [–t` *char* `] [–k` *keydef* `]` . . . [*file* . . .]

`sort –c [–bdfinru] [–t` *char* `] [–k` *keydef* `]` . . . [*file*]

Obsolescent Versions :

`sort [–mu] [–o` *output* `] [–bdfinr] [–t` *char* `]` [+*pos1* [–*pos2*]] . . . [*file* . . .]

`sort –c [–u] [–bdfinr] [–t` *char* `]` [+*pos1* [–*pos2*]] . . . [*file*]

4.58.2 Description

The `sort` utility shall perform one of the following functions:

(1) Sort lines of all the named files together and write the result to the specified output.

(2) Merge lines of all the named (presorted) files together and write the result to the specified output.

(3) Check that a single input file is correctly presorted.

Comparisons shall be based on one or more sort keys extracted from each line of input (or the entire line if no sort keys are specified), and shall be performed using the collating sequence of the current locale.

4.58.3 Options

The `sort` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the notation +*pos1* −*pos2* uses a nonstandard prefix and multidigit option names in the obsolescent versions, the `−o` *output* option shall be recognized after a *file* operand as an obsolescent feature in both versions where the `−c` option is not specified, and the `−k` *keydef* option should follow the `−b`, `−d`, `−f`, `−i`, `−n`, and `−r` options.

The following options shall be supported by the implementation:

`−c` Check that the single input file is ordered as specified by the arguments and the collating sequence of the current locale. No output shall be produced; only the exit code shall be affected.

`−m` Merge only; the input files shall be assumed to be already sorted.

`−o` *output* Specify the name of an output file to be used instead of the standard output. This file can be the same as one of the input *file* s.

`−u` Unique: suppress all but one in each set of lines having equal keys. If used with the `−c` option, check that there are no lines with duplicate keys, in addition to checking that the input file is sorted.

The following options shall override the default ordering rules. When ordering options appear independent of any key field specifications, the requested field ordering rules shall be applied globally to all sort keys. When attached to a specific key (see `−k`), the specified ordering options shall override all global ordering options for that key. In the obsolescent forms, if one or more of these options follows a +*pos1* option, it shall affect only the key field specified by that preceding option.

`−d` Specify that only `<blank>`s and alphanumeric characters, according to the current setting of LC_CTYPE , shall be significant in comparisons. The behavior is undefined for a sort key to which `−i` or `−n` also applies.

`−f` Consider all lowercase characters that have uppercase equivalents, according to the current setting of LC_CTYPE , to be the uppercase equivalent for the purposes of comparison.

`−i` Ignore all characters that are nonprintable, according to the current setting of LC_CTYPE.

−n Restrict the sort key to an initial numeric string, consisting of optional `<blank>`s, optional minus sign, and zero or more digits with an optional radix character and thousands separators (as defined in the current locale), which shall be sorted by arithmetic value. An empty digit string shall be treated as zero. Leading zeros and signs on zeros shall not affect ordering.

−r Reverse the sense of comparisons.

The treatment of field separators can be altered using the options:

−b Ignore leading `<blank>`s when determining the starting and ending positions of a restricted sort key. If the −b option is specified before the first −k option, it shall be applied to all −k options. Otherwise, the −b option can be attached independently to each −k *field_start* or *field_end* option-argument (see below).

−t *char* Use *char* as the field separator character; *char* shall not be considered to be part of a field (although it can be included in a sort key). Each occurrence of *char* shall be significant (for example, *<char><char>* shall delimit an empty field). If −t is not specified, `<blank>` characters shall be used as default field separators; each maximal nonempty sequence of `<blank>` characters that follows a non-`<blank>` character shall be a field separator.

Sort keys can be specified using the options:

−k *keydef* The *keydef* argument is a restricted sort key field definition. The format of this definition is

field_start [*type*][,*field_end* [*type*]]

where *field_start* and *field_end* define a key field restricted to a portion of the line (see 4.58.7), and *type* is a modifier from the list of characters b, d, f, i, n, r. The b modifier shall behave like the −b option, but applies only to the *field_start* or *field_end* to which it is attached. The other modifiers shall behave like the corresponding options, but shall apply only to the key field to which they are attached; they shall have this effect if specified with *field_start,* *field_end,* or both. Modifiers attached to a *field_start* or *field_end* shall override any specifications made by the options. Implementations shall support at least nine occurrences of the −k option, which shall be significant in command line order. If no −k option is specified, a default sort key of the entire line shall be used.

When there are multiple key fields, later keys shall be compared only after all earlier keys compare equal. Except when the −u option is specified, lines that otherwise compare equal shall be ordered as if none of the options −d, −f, −i, −n, or −k were present (but with −r still in effect, if it was specified) and with all bytes in the lines significant to the comparison. The order in which lines that still compare equal are written is unspecified.

+pos1 (Obsolescent.) Specify the start position of a key field. See 4.58.7.

−pos2 (Obsolescent.) Specify the end position of a key field. See 4.58.7.

4.58.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file to be sorted, merged, or checked. If no *file* operands are specified, or if a *file* operand is −, the standard input shall be used.

4.58.5 External Influences

4.58.5.1 Standard Input

The standard input shall be used only if no *file* operands are specified, or if a *file* operand is −. See Input Files.

4.58.5.2 Input Files

The input files shall be text files, except that the sort utility shall add a <newline> to the end of a file ending with an incomplete last line.

4.58.5.3 Environment Variables

The following environment variables shall affect the execution of sort:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_COLLATE This variable shall determine the locale for ordering rules.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files) and the behavior of character classification for the −b, −d, −f, −i, and −n options.

LC_MESSAGES This variable shall determine the language in which messages should be written.

LC_NUMERIC This variable shall determine the locale for the definition of the radix character and thousands separator for the `-n` option.

4.58.5.4 Asynchronous Events

Default.

4.58.6 External Effects

4.58.6.1 Standard Output

Unless the `-o` or `-c` options are in effect, the standard output shall contain the sorted input.

4.58.6.2 Standard Error

Used only for diagnostic messages. A warning message about correcting an 2
incomplete last line of an input file may be generated, but need not affect the final 2
exit status. 2

4.58.6.3 Output Files

If the `-o` option is in effect, the sorted input shall be placed in the file *output*.

4.58.7 Extended Description

The notation

`-k` *field_start* [*type*][`,`*field_end* [*type*]]

shall define a key field that begins at *field_start* and ends at *field_end* inclusive, unless *field_start* falls beyond the end of the line or after *field_end*, in which case the key field shall be empty. A missing *field_end* shall mean the last character of the line.

A field comprises a maximal sequence of nonseparating characters and, in the 1
absence of option `-t`, any preceding field separator. 1

The *field_start* portion of the *keydef* option argument shall have the form:

field_number [`.`*first_character*]

Fields and characters within fields shall be numbered starting with 1. The *field_number* and *first_character* pieces, interpreted as positive decimal integers, shall specify the first character to be used as part of a sort key. If `.`*first_character* is omitted, it shall refer to the first character of the field.

The *field_end* portion of the *keydef* option argument shall have the form:

field_number [.*last_character*]

The *field_number* shall be as described above for *field_start*. The *last_character* piece, interpreted as a nonnegative decimal integer, shall specify the last character to be used as part of the sort key. If *last_character* evaluates to zero or .*last_character* is omitted, it shall refer to the last character of the field specified by *field_number*.

If the −b option or b type modifier is in effect, characters within a field shall be counted from the first non-<blank> in the field. (This shall apply separately to *first_character* and *last_character* .)

The obsolescent [+*pos1* [−*pos2*]] options provide functionality equivalent to the −k *keydef* option. For comparison, the full formats of these options shall be:

+*field0_number* [.*first0_character*][*type*] [−*field0_number* [.*first0_character*][*type*]]
-k *field_number* [.*first_character*][*type*][,*field_number* [.*last_character*][*type*]]

In the obsolescent form, fields (specified by *field0_number*) and characters within fields (specified by *first0_character*) shall be numbered from zero instead of one. The −*pos2* option shall specify the first character after the sort field instead of the last character in the sort field. (Therefore, *field0_number* and *first0_character* shall be interpreted as nonnegative, instead of positive, decimal integers and there is no need for a specification of a *last_character* -like form.) The optional type modifiers shall be the same in both forms. If .*first0_character* is omitted or *first0_character* evaluates to zero, it shall refer to the first character of the field.

Thus, a the fully specified +*pos1* −*pos2* form:

+*w*.*x* -*y*.*z*

shall be equivalent to:

-k *w*+1.*x*+1,*y*.0 (if *z* == 0)
-k *w*+1.*x*+1,*y*+1.*z* (if *z* > 0)

As with the nonobsolescent forms, implementations shall support at least nine occurrences of the +*pos1* option, which shall be significant in command line order.

4.58.8 Exit Status

The sort utility shall exit with one of the following values:

0 All input files were output successfully, or −c was specified and the input file was correctly sorted.

1 Under the −c option, the file was not ordered as specified, or if the −c and −u options were both specified, two input lines were found with equal keys. This exit status shall not be returned if the −c option is not used.

>1 An error occurred.

4.58.9 Consequences of Errors

Default.

4.58.10 Rationale. *(This subclause is not a part of P1003.2)*

Examples, Usage

In the following examples, nonobsolescent and obsolescent ways of specifying sort keys are given as an aid to understanding the relationship between the two forms.

Either of the following commands sorts the contents of infile with the second field as the sort key:

```
sort -k 2,2 infile
sort +1 -2 infile
```

Either of the following commands sorts, in reverse order, the contents of infile1 and infile2, placing the output in outfile and using the second character of the second field as the sort key (assuming that the first character of the second field is the field separator):

```
sort -r -o outfile -k 2.2,2.2 infile1 infile2
sort -r -o outfile +1.1 -1.2 infile1 infile2
```
1

Either of the following commands sorts the contents of infile1 and infile2 using the second non-<blank> character of the second field as the sort key:

```
sort -k 2.2b,2.2b infile1 infile2
sort +1.1b -1.2b  infile1 infile2
```

Either of the following commands prints the System V password file (user database) sorted by the numeric user ID (the third colon-separated field):

```
sort -t : -k 3,3n /etc/passwd
sort -t : +2 -3n /etc/passwd
```

Either of the following commands prints the lines of the already sorted file infile, suppressing all but one occurrence of lines having the same third field:

```
sort -um -k 3.1,3.0 infile
sort -um +2.0 -3.0 infile
```

Examples in some historical documentation state that options –um with one input 1
file keep the first in each set of lines with equal keys. This behavior was deemed 2
to be an implementation artifact and was not made standard. 1

The default value for –t, <blank>, has different properties than, for example, -t "<space>". If a line contains:

```
<space><space>foo
```

the following treatment would occur with default separation versus specifically selecting a <space>:

Field	Default	`-t "<space>"`
1	`<space><space>foo`	*empty*
2	*empty*	*empty*
3	*empty*	`foo`

The leading field separator itself is included in a field when `-t` is not used. For example, this command returns an exit status of zero, meaning the input was already sorted:

```
sort -c -k 2 <<eof
y<tab>b
x<space>a
eof
```

(assuming that `<tab>` precedes `<space>` in the current collating sequence). The field separator is not included in a field when it is explicitly set via `-t`. This is historical practice and allows usage such as

```
sort -t "|" -k 2n <<eof
Atlanta|425022|Georgia
Birmingham|284413|Alabama
Columbia|100385|South Carolina
eof
```

where the second field can be correctly sorted numerically without regard to the nonnumeric field separator.

History of Decisions Made

The `-z` option was removed; it is not standard practice on most systems, and is inconsistent with using `sort` to individually sort several files and then merging them together. The previous language appeared to require implementations to determine the proper buffer length during the sort phase of operation, but not during the merge.

The `-y` option was removed because of nonportability. The `-M` option, present in System V, was removed because of nonportability in international usage.

An undocumented `-T` option exists in some implementations. It is used to specify a directory for intermediate files. Implementations are encouraged to support the use of the **TMPDIR** environment variable instead of adding an option to support this functionality.

The `-k` option was added to satisfy two complaints. First, the zero-based counting used by `sort` is not consistent with other utility conventions. Second, it did not meet syntax guideline requirements. The one-based counting in this standard was developed from the input provided by several ballot comments, ballot objections, and discussions with users.

The wording in Draft 10 also clarifies that the `-b`, `-d`, `-f`, `-i`, `-n`, and `-r` options have to come before the first sort key specified if they are intended to apply to all specified keys. The way it is described in this standard matches historical practice, not historical documentation. In the nonobsolescent versions, the results are unspecified if these options are specified after a −k option. This will allow implementations to make the options independent of each other when the obsolescent forms are finally dropped (if that ever happens).

Historical documentation indicates that ‘‘setting −n implies −b.’’ The description of −n already states that optional leading <blank>s are tolerated in doing the comparison. If −b is enabled, rather than implied, by −n, this has unusual side effects. When a character offset is used into a column of numbers (e.g., to sort mod 100), that offset will be measured relative to the most significant digit, not to the column. Based upon a recommendation of the author of the original `sort` utility, the −b implication has been omitted from POSIX.2 and an application wishing to achieve the previously mentioned side effects will have to manually code the −b flag.

4.59 `stty` — Set the options for a terminal

4.59.1 Synopsis

`stty [ −a | −g ]`

`stty` *operands*

4.59.2 Description

The `stty` utility shall set or report on terminal I/O characteristics for the device that is its standard input. Without options or operands specified, it shall report the settings of certain characteristics, usually those that differ from implementation-defined defaults. Otherwise, it shall modify the terminal state according to the specified operands. Detailed information about the modes listed in the first five groups below are described in POSIX.1 {8} Section 7. Operands in the Combination Modes group (see 4.59.4.6) shall be implemented using operands in the previous groups. Some combinations of operands are mutually exclusive on some terminal types; the results of using such combinations are unspecified.

Typical implementations of this utility require a communications line configured to use a POSIX.1 {8} *termios* interface. On systems where none of these lines are available, and on lines not currently configured to support the POSIX.1 {8} termios interface, some of the operands need not affect terminal characteristics.

4.59.3 Options

The `stty` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

`−a` Write to standard output all the current settings for the terminal.

`−g` Write to standard output all the current settings in an unspecified form that can be used as arguments to another invocation of the `stty` utility on the same system. The form used shall not contain any characters that would require quoting to avoid word expansion by the shell; see 3.6.

4.59.4 Operands

The following operands shall be supported by the implementation to set the terminal characteristics:

4.59.4.1 Control Modes

`parenb (−parenb)` Enable (disable) parity generation and detection. This shall have the effect of setting (not setting) PARENB in the *termios c_cflag* field, as defined in POSIX.1 {8}.

`parodd (−parodd)` Select odd (even) parity. This shall have the effect of setting (not setting) PARODD in the *termios c_cflag* field, as defined in POSIX.1 {8}.

`cs5 cs6 cs7 cs8` Select character size, if possible. This shall have the effect of setting CS5, CS6, CS7, and CS8, respectively, in the *termios c_cflag* field, as defined in POSIX.1 {8}.

number Set terminal baud rate to the number given, if possible. If the baud rate is set to zero, the modem control lines shall no longer be asserted. This shall have the effect of setting the input and output *termios* baud rate values as defined in POSIX.1 {8}.

`ispeed` *number* Set terminal input baud rate to the number given, if possible. If the input baud rate is set to zero, the input baud rate shall be specified by the value of the output baud rate. This shall have the effect of setting the input *termios* baud rate values as defined in POSIX.1 {8}.

`ospeed` *number* Set terminal output baud rate to the number given, if possible. If the output baud rate is set to zero, the modem control lines shall no longer be asserted. This shall have the effect of setting the output *termios* baud rate values as defined in POSIX.1 {8}.

`hupcl (−hupcl)` Stop asserting modem control lines (do not stop asserting modem control lines) on last close. This shall have the effect of setting (not setting) HUPCL in the *termios c_cflag* field, as defined in POSIX.1 {8}.

`hup (−hup)` Same as `hupcl` (`−hupcl`).

`cstopb (−cstopb)` Use two (one) stop bits per character. This shall have the effect of setting (not setting) CSTOPB in the *termios c_cflag* field, as defined in POSIX.1 {8}.

`cread (−cread)` Enable (disable) the receiver. This shall have the effect of setting (not setting) CREAD in the *termios c_cflag* field, as defined in POSIX.1 {8}.

`clocal (−clocal)` Assume a line without (with) modem control. This shall have the effect of setting (not setting) CLOCAL in the *termios c_cflag* field, as defined in POSIX.1 {8}.

It is unspecified whether `stty` shall report an error if an attempt to set a Control Mode fails.

4.59.4.2 Input Modes

`ignbrk (−ignbrk)` Ignore (do not ignore) break on input. This shall have the effect of setting (not setting) IGNBRK in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`brkint (−brkint)` Signal (do not signal) INTR on break. This shall have the effect of setting (not setting) BRKINT in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`ignpar (−ignpar)` Ignore (do not ignore) bytes with parity errors. This shall have the effect of setting (not setting) IGNPAR in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`parmrk (−parmrk)` Mark (do not mark) parity errors. This shall have the effect of setting (not setting) PARMRK in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`inpck (−inpck)` Enable (disable) input parity checking. This shall have the effect of setting (not setting) INPCK in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`istrip (−istrip)` Strip (do not strip) input characters to seven bits. This shall have the effect of setting (not setting) ISTRIP in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`inlcr (−inlcr)` Map (do not map) NL to CR on input. This shall have the effect of setting (not setting) INLCR in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`igncr (−igncr)` Ignore (do not ignore) CR on input. This shall have the effect of setting (not setting) IGNCR in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`icrnl (−icrnl)` Map (do not map) CR to NL on input. This shall have the effect of setting (not setting) ICRNL in the *termios* *c_iflag* field, as defined in POSIX.1 {8}.

`ixon (−ixon)` Enable (disable) START/STOP output control. Output from the system is stopped when the system receives STOP and started when the system receives START. This shall have the effect of setting (not setting) IXON in the *termios c_iflag* field, as defined in POSIX.1 {8}.

`ixoff (−ixoff)` Request that the system send (not send) STOP characters when the input queue is nearly full and START characters to resume data transmission. This shall have the effect of setting (not setting) IXOFF in the *termios c_iflag* field, as defined in POSIX.1 {8}.

4.59.4.3 Output Modes

`opost (−opost)` Post-process output (do not post-process output; ignore all other output modes). This shall have the effect of setting (not setting) OPOST in the *termios c_oflag* field, as defined in POSIX.1 {8}.

4.59.4.4 Local Modes

`isig (−isig)` Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SUSP. This shall have the effect of setting (not setting) ISIG in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`icanon (−icanon)` Enable (disable) canonical input (ERASE and KILL processing). This shall have the effect of setting (not setting) ICANON in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`iexten (−iexten)` Enable (disable) any implementation-defined special control characters not currently controlled by `icanon`, `isig`, `ixon`, or `ixoff`. This shall have the effect of setting (not setting) IEXTEN in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`echo (−echo)` Echo back (do not echo back) every character typed. This shall have the effect of setting (not setting) ECHO in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`echoe (−echoe)` The ERASE character shall (shall not) visually erase the last character in the current line from the display, if possible. This shall have the effect of setting (not setting) ECHOE in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`echok (−echok)` Echo (do not echo) NL after KILL character. This shall have the effect of setting (not setting) ECHOK in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`echonl (−echonl)` Echo (do not echo) NL, even if `echo` is disabled. This shall have the effect of setting (not setting) ECHONL in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`noflsh (−noflsh)` Disable (enable) flush after INTR, QUIT, SUSP. This shall have the effect of setting (not setting) NOFLSH in the *termios c_lflag* field, as defined in POSIX.1 {8}.

`tostop (−tostop)` Send SIGTTOU for background output. This shall 2
have the effect of setting (not setting) TOSTOP in the 2
termios c_lflag field, as defined in POSIX.1 {8}. 2

NOTE: Setting TOSTOP has no effect on systems not supporting 2
the POSIX.1 {8} job control option. 2

4.59.4.5 Special Control Character Assignments

control-character string

Set *control-character* to *string*. If *control-character* is one of the character sequences in the first column of Table 4-9, the corresponding POSIX.1 {8} control character from the second column shall be recognized. This shall have the effect of setting the corresponding element of the *termios c_cc* array (see POSIX.1 {8} 7.1.2).

Table 4-9 − `stty` Control Character Names

control-character	POSIX.1 {8} Subscript	Description
`eof`	VEOF	EOF character
`eol`	VEOL	EOL character
`erase`	VERASE	ERASE character
`intr`	VINTR	INTR character
`kill`	VKILL	KILL character
`quit`	VQUIT	QUIT character
`susp`	VSUSP	SUSP character
`start`	VSTART	START character
`stop`	VSTOP	STOP character

If *string* is a single character, the control character shall be set to that character. If *string* is the two-character sequence `"^−"` or the string `"undef"`, the control character shall be set to {_POSIX_VDISABLE}, if it is in effect for the device; if {_POSIX_VDISABLE} is not in effect for the device, it shall be treated as an error. In the POSIX Locale, if *string* is a two-character sequence beginning with circumflex (^), and the second character is one of those listed in the *^c* column of Table 4-10, the control character shall be set to the corresponding character value in the Value column of the table.

Table 4-10 – `stty` Circumflex Control Characters

^c	**Value**	*^c*	**Value**	*^c*	**Value**
`a, A`	`<SOH>`	`l, L`	`<FF>`	`w, W`	`<ETB>`
`b, B`	`<STX>`	`m, M`	`<CR>`	`x, X`	`<CAN>`
`c, C`	`<ETX>`	`n, N`	`<SO>`	`y, Y`	`<EM>`
`d, D`	`<EOT>`	`o, O`	`<SI>`	`z, Z`	`<SUB>`
`e, E`	`<ENQ>`	`p, P`	`<DLE>`	`[`	`<ESC>`
`f, F`	`<ACK>`	`q, Q`	`<DC1>`	`\`	`<FS>`
`g, G`	`<BEL>`	`r, R`	`<DC2>`	`]`	`<GS>`
`h, H`	`<BS>`	`s, S`	`<DC3>`	`^`	`<RS>`
`i, I`	`<HT>`	`t, T`	`<DC4>`	`_`	`<US>`
`j, J`	`<LF>`	`u, U`	`<NAK>`	`?`	`<DEL>`
`k, K`	`<VT>`	`v, V`	`<SYN>`		

`min` *number*
`time` *number* — Set the value of `min` or `time` to *number*. MIN and TIME are used in noncanonical mode input processing (`–icanon`).

4.59.4.6 Combination Modes

saved settings — Set the current terminal characteristics to the saved settings produced by the `–g` option.

`evenp` or `parity` — Enable `parenb` and `cs7`; disable `parodd`.

`oddp` — Enable `parenb`, `cs7`, and `parodd`.

`–parity`, `–evenp`, or `–oddp`
Disable `parenb`, and set `cs8`.

`nl (–nl)` — Enable (disable) `icrnl`. In addition, `–nl` unsets `inlcr` and `igncr`.

`ek` — Reset ERASE and KILL characters back to system defaults.

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

4.59.5 External Influences

4.59.5.1 Standard Input

Although no input is read from standard input, standard input is used to get the current terminal I/O characteristics and to set new terminal I/O characteristics.

4.59.5.2 Input Files

None.

4.59.5.3 Environment Variables

The following environment variables shall affect the execution of `stty`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments) and which characters are in the class `print`.

LC_MESSAGES This variable shall determine the language in which messages should be written.

4.59.5.4 Asynchronous Events

Default.

4.59.6 External Effects

4.59.6.1 Standard Output

If operands are specified, no output shall be produced.

If the `−g` option is specified, `stty` shall write to standard output the current settings in a form that can be used as arguments to another instance of `stty` on the same system.

If the `−a` option is specified, all of the information as described in 4.59.4 shall be written to standard output. Unless otherwise specified, this information shall be written as `<space>`-separated tokens in an unspecified format, on one or more lines, with an unspecified number of tokens per line. Additional information may be written.

If no options or operands are specified, an unspecified subset of the information written for the `−a` option shall be written.

If speed information is written as part of the default output, or if the `−a` option is specified and if the terminal input speed and output speed are the same, the speed information shall be written as follows:

`"speed %d baud;"`, *<speed>*

Otherwise, speeds shall be written as:

`"ispeed %d baud; ospeed %d baud;"`, *<ispeed>*, *<ospeed>*

In locales other than the POSIX Locale, the word `baud` may be changed to something more appropriate in those locales.

If control characters are written as part of the default output, or if the `−a` option is specified, control characters shall be written as:

`"%s = %s;"`, *<control-character name>*, *<value>*

where *value* is either the character, or some visual representation of the character if it is nonprintable, or the string `<undef>` if the character is disabled.

4.59.6.2 Standard Error

Used only for diagnostic messages.

4.59.6.3 Output Files

None.

4.59.7 Extended Description

None.

4.59.8 Exit Status

The `stty` utility shall exit with one of the following values:

- 0 The terminal options were read or set successfully.
- >0 An error occurred.