-- --

IEEE P1003.2 Draft 11.2 – September 1991

`−c` Do not create a specified *file* if it does not exist. Do not write any diagnostic messages concerning this condition.

`−m` Change the modification time of *file*. Do not change the access time unless `−a` is also specified.

`−r` *ref_file* Use the corresponding time of the file named by the pathname *ref_file* instead of the current time.

`−t` *time* Use the specified *time* instead of the current time. The option-argument shall be a decimal number of the form:

[[*CC*]*YY*]*MMDD hhmm*[`.`*SS*]

where each two digits represents the following:

MM	The month of the year (01-12).
DD	The day of the month (01-31).
hh	The hour of the day (00-23).
mm	The minute of the hour (00-59).
CC	The first two digits of the year (the century).
YY	The second two digits of the year.
SS	The second of the minute (00-61).

Both *CC* and *YY* shall be optional. If neither is given, the current year shall be assumed. If *YY* is specified, but *CC* is not, *CC* shall be derived as follows:

If *YY* is:	*CC* becomes:
69-99	19
00-68	20

The resulting time shall be affected by the value of the **TZ** environment variable. If the resulting time value precedes the Epoch, `touch` shall exit immediately with an error status. The range of valid times past the Epoch is implementation defined, but shall extend to at least midnight 1 January 2000 UTC.

The range for *SS* is (00-61) rather than (00-59) because of leap seconds. If *SS* is 60 or 61, and the resulting time, as affected by the **TZ** environment variable, does not refer to a leap second: the resulting time shall be one or two seconds after a time where *SS* is 59. If *SS* is not given a value, it is assumed to be zero.

If neither the `−a` nor `−m` options were specified, `touch` shall behave as if both the `−a` and `−m` options were specified.

4.63.4 Operands

The following operands shall be supported by the implementation:

file A pathname of a file whose times are to be modified.

date_time (Obsolescent.) Use the specified *date_time* instead of the current time. The operand is a decimal number of the form:

MMDDhhmm[*yy*]

where *MM*, *DD*, *hh*, and *mm* are as described for the *time* option-argument to the `-t` option and the optional *yy* is interpreted as follows:

If not specified, the current year shall be used. If *yy* is in the range 69-99, the year 1969-1999, respectively, shall be used. Otherwise, the results are unspecified.

If no `-r` option is specified, no `-t` option is specified, at least two operands are specified, and the first operand is an eight- or ten-digit decimal integer, the first operand shall be assumed to be a *date_time* operand. Otherwise, the first operand shall be assumed to be a *file* operand.

4.63.5 External Influences

4.63.5.1 Standard Input

None.

4.63.5.2 Input Files

None.

4.63.5.3 Environment Variables

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

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.

TZ If the *time* option-argument (or operand; see above) is specified, **TZ** shall be used to interpret the time for the specified time zone.

4.63.5.4 Asynchronous Events

Default.

4.63.6 External Effects

4.63.6.1 Standard Output

None.

4.63.6.2 Standard Error

Used only for diagnostic messages.

4.63.6.3 Output Files

None.

4.63.7 Extended Description

None.

4.63.8 Exit Status

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

0 The utility executed successfully and all requested changes were made.

>0 An error occurred.

4.63.9 Consequences of Errors

Default.

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

Examples, Usage

The functionality of `touch` is described almost entirely through references to functions in POSIX.1 {8}. In this way, there is no duplication of effort required for describing such side effects as the relationship of user IDs to the user database, permissions, etc.

The interpretation of time is taken to be ''seconds since the Epoch,'' as defined by 2.2.2.129. It should be noted that POSIX.1 {8} conforming implementations do not take leap seconds into account when computing seconds since the Epoch. When *SS* =60 is used on POSIX.1 {8} conforming implementations, the resulting time always refers to 1 plus ''seconds since the Epoch'' for a time when *SS* =59.

Note that although the `-t` *time* option-argument and the obsolescent *date_time* operand specify values in 1969, the access time and modification time fields are defined in terms of seconds since the Epoch (midnight on 1 January 1970 UTC). 1 Therefore, depending on the value of **TZ** when `touch` is run, there will never be more than a few valid hours in 1969 and there need not be any valid times in 1969.

History of Decisions Made

There are some significant differences between this `touch` and those in System V and BSD systems. They are upward compatible for existing applications from both implementations.

(1) In System V, an ambiguity exists when a pathname that is a decimal number leads the operands; it is treated as a time value. In BSD, no *time* value is allowed; files may only be `touch`ed to the current time. The [`-t` *time*] construct solves these problems for future portable applications (note that the `-t` option is not existing practice).

(2) The inclusion of the century digits, *CC*, is also new. Note that a ten-digit *time* value is treated as if *YY*, and not *CC*, were specified. The caveat about the range of dates following the Epoch was included as recognition that some UNIX systems will not be able to represent dates beyond the January 18, 2038, because they use *signed int* as a time holder.

One ambiguous situation occurs if `-t` *time* is not specified, `-r` *ref_file* is not specified, and the first operand is an eight- or ten-digit decimal number. A portable script can avoid this problem by using:

```
touch -- file
```

or

```
touch ./file
```

in this case.

The `-r` option was added because several comments requested this capability. This option was named `-f` in an earlier draft, but was changed because the `-f` option is used in the BSD version of `touch` with a different meaning.

At least one historical implementation of touch incremented the exit code if `-c` was specified and the file did not exist. This standard requires exit status zero if no errors occur.

4.64 `tr` — Translate characters

4.64.1 Synopsis

`tr [-cs]` *string1 string2*

`tr -s [-c]` *string1*

`tr -d [-c]` *string1*

`tr -ds [-c]` *string1 string2*

4.64.2 Description

The `tr` utility shall copy the standard input to the standard output with substitution or deletion of selected characters. The options specified and the *string1* and *string2* operands shall control translations that occur while copying characters and collating elements.

4.64.3 Options

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

The following options shall be supported by the implementation:

`-c` Complement the set of characters specified by *string1* . See 4.64.7.

`-d` Delete all occurrences of input characters that are specified by *string1* .

`-s` Replace instances of repeated characters with a single character, as described in 4.64.7. 1

4.64.4 Operands

The following operands shall be supported by the implementation:

string1
string2 Translation control strings. Each string shall represent a set of characters to be converted into an array of characters used for the translation. For a detailed description of how the strings are interpreted, see 4.64.7.

4.64.5 External Influences

4.64.5.1 Standard Input

The standard input can be any type of file.

4.64.5.2 Input Files

None.

4.64.5.3 Environment Variables

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

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 behavior of range expressions and equivalence classes.

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 the behavior of character classes.

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

4.64.5.4 Asynchronous Events

Default.

4.64.6 External Effects

4.64.6.1 Standard Output

The `tr` output shall be identical to the input, with the exception of the specified transformations.

4.64.6.2 Standard Error

Used only for diagnostic messages.

4.64.6.3 Output Files

None.

4.64.7 Extended Description

The operands *string1* and *string2* (if specified) define two arrays of characters or collating elements. The following conventions can be used to specify characters or collating elements:

character Any character not described by one of the conventions below shall represent itself.

octal Octal sequences can be used to represent characters with specific coded values. An octal sequence shall consist of a backslash followed by the longest sequence of one-, two-, or three-octal-digit characters (01234567). The sequence shall cause the character whose encoding is represented by the one-, two-, or three-digit octal integer to be placed into the array. 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. Multibyte characters require multiple, concatenated escape sequences of this type, including the leading \ for each byte.

character The backslash-escape sequences in Table 2-15 (see 2.12) shall be supported. The results of using any other character, other than an octal digit, following the backslash are unspecified.

c−*c* Represents the range of collating elements between the range endpoints, inclusive, as defined by the current setting of the LC_COLLATE locale category. The starting endpoint shall precede the second endpoint in the current collation order. The characters or collating elements in the range shall be placed in the array in ascending collation sequence. No multicharacter collating elements shall be included in the range.

`[:`*class*`:]` Represents all characters belonging to the defined character class, as defined by the current setting of the LC_CTYPE locale category. The following character class names shall be accepted when specified in *string1* :

```
alnum   cntrl   lower   space
alpha   digit   print   upper
blank   graph   punct   xdigit
```

When the `-d` and `-s` options are specified together, any of the character class names shall be accepted in *string2* . Otherwise, only character class names `lower` or `upper` shall be accepted in *string2* and then only if the corresponding character class (`upper` and `lower`, respectively) is specified in the same relative position in *string1* . Such a specification shall be interpreted as a request for case conversion. When `[:lower:]` appears in *string1* and `[:upper:]` appears in *string2,* the arrays shall contain the characters from the `toupper` mapping in the LC_CTYPE category of the current locale. When `[:upper:]` appears in *string1* and `[:lower:]` appears in *string2,* the arrays shall contain the characters from the `tolower` mapping in the LC_CTYPE category of the current locale. The first character from each mapping pair shall be in the array for *string1* and the second character from each mapping pair shall be in the array for *string2* in the same relative position.

Except for case conversion, the characters specified by a character class expression shall be placed in the array in an unspecified order.

If the name specified for *class* does not define a valid character class in the current locale, the behavior is undefined.

[=*equiv* =] Represents all characters or collating elements belonging to the same equivalence class as *equiv* , as defined by the current setting of the LC_COLLATE locale category. An equivalence class expression shall be allowed only in *string1* , or in *string2* when it is being used by the combined `-d` and `-s` options. The characters belonging to the equivalence class shall be placed in the array in an unspecified order.

[*x***n*] Represents *n* repeated occurrences of the character or collating symbol *x*. Because this expression is used to map multiple characters to one, it is only valid when it occurs in *string2* . If *n* is omitted or is zero, it shall be interpreted as large enough to extend the *string2* -based sequence to the length of the *string1* -based sequence. If *n* has a leading zero, it shall be interpreted as an octal value. Otherwise, it shall be interpreted as a decimal value.

When the `-d` option is not specified:

— Each input character or collating element found in the array specified by *string1* shall be replaced by the character or collating element in the same relative position in the array specified by *string2* . When the array specified by *string2* is shorter that the one specified by *string1* , the results are unspecified.

— If the `-c` option is specified without `-d`, the complement of the characters specified by *string1* —the set of all characters in the current character set, as defined by the current setting of LC_CTYPE , except for those actually specified in the *string1* operand—shall be placed in the array in ascending collation sequence, as defined by the current setting of LC_COLLATE.

— Because the order in which characters specified by character class expressions or equivalence class expressions is undefined, such expressions should only be used if the intent is to map several characters into one. An exception is case conversion, as described previously.

When the `-d` option is specified:

— Input characters or collating elements found in the array specified by *string1* shall be deleted.

— When the `-c` option is specified with `-d`, all characters except those specified by *string1* shall be deleted. The contents of *string2* shall be ignored, unless the `-s` option is also specified.

— The same string cannot be used for both the `-d` and the `-s` option; when both options are specified, both *string1* (used for deletion) and *string2* (used for squeezing) shall be required.

When the `-s` option is specified, after any deletions or translations have taken place, repeated sequences of the same character shall be replaced by one occurrence of the same character, if the character is found in the array specified by the last operand. If the last operand contains a character class, such as the following example:

```
tr -s '[:space:]'
```

the last operand's array shall contain all of the characters in that character class. However, in a case conversion, as described previously, such as

```
tr -s '[:upper:]' '[:lower:]'
```

the last operand's array shall contain only those characters defined as the second characters in each of the `toupper` or `tolower` character pairs, as appropriate.

4.64.8 Exit Status

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

- 0 All input was processed successfully.
- >0 An error occurred.

4.64.9 Consequences of Errors

Default.

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

Examples, Usage

If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

The following example creates a list of all words in *file1* one per line in *file2*, where a word is taken to be a maximal string of letters.

```
tr -cs "[:alpha:]" "[\n*]" <file1 >file2
```

If an ordinary digit (representing itself) is to follow an octal sequence, the octal sequence must use the full three digits to avoid ambiguity.

When *string2* is shorter than *string1*, a difference results between historical System V and BSD systems. A BSD system will pad *string2* with the last character found in *string2*. Thus, it is possible to do the following:

```
tr 0123456789 d
```

which would translate all digits to the letter `d`. Since this area is specifically unspecified in the standard, both the BSD and System V behaviors are allowed, but a conforming application cannot rely on the BSD behavior. It would have to code the example in the following way:

```
tr 0123456789 '[d*]'
```

It should be noted that, despite similarities in appearance, the string operands used by `tr` are not regular expressions.

On historical System V systems, a range expression requires enclosing square- 2
brackets, such as: 2

```
tr '[a-z]' '[A-Z]'
```
2

However, BSD-based systems did not require the brackets and this convention is 2
used by POSIX.2 to avoid breaking large numbers of BSD scripts: 2

```
tr a-z A-Z
```
2

The preceding System V script will continue to work because the brackets, treated 2
as regular characters, are translated to themselves. However, any System V 2

script that relied on `a-z` representing the three characters `a`, `-`, and `z` will have to 2
be rewritten as `az-` or `a\-z`. 2

History of Decisions Made

In some earlier drafts, an explicit option, `–n`, was added to disable the historical behavior of stripping NUL characters from the input. It was felt that automatically stripping NUL characters from the input was not correct functionality. However, the removal of `–n` in a later draft does not remove the requirement that `tr` correctly process NUL characters in its input stream. NUL characters can be stripped by using `tr –d '\000'`.

Historical implementations of `tr` differ widely in syntax and behavior. For example, the BSD version has not needed the bracket characters for the repetition sequence. The POSIX.2 `tr` syntax is based more closely on the System V and XPG3 model, while attempting to accommodate historical BSD implementations. In the case of the short *string2* padding, the decision was to unspecify the behavior and preserve System V and XPG scripts, which might find difficulty with the BSD method. The assumption was made that BSD users of `tr` will have to make accommodations to meet the POSIX.2 syntax anyway, and since it is possible to use the repetition sequence to duplicate the desired behavior, whereas there is no simple way to achieve the System V method, this was the correct, if not desirable, approach.

The use of octal values to specify control characters, while having historical precedents, is not portable. The introduction of escape sequences for control characters should provide the necessary portability. It is recognized that this may cause some historical scripts to break.

A previous draft included support for multicharacter collating elements. Several balloters pointed out that, while `tr` does employ some syntactical elements from regular expressions, the aim of `tr` is quite different; ranges, for instance, do not mean the same thing (''any of the chars in the range matches,'' versus ''translate each character in the range to the output counterpart''). As a result, the previously included support for multicharacter collating elements has been removed. What remains are ranges in current collation order (to support, e.g., accented characters), character classes, and equivalence classes.

In XPG3, the `[:class:]` and `[=equiv=]` conventions are shown with double brackets, as in regular expression syntax. Several balloters objected to this, pointing out that `tr` does not implement regular expression principles, just borrows part of the syntax. Consequently, the `[:class:]` and `[=equiv=]` should be regarded as syntactical elements on a par with `[x*n]`, which is not an RE bracket expression.

4.65 `true` — Return true value

4.65.1 Synopsis

```
true
```

4.65.2 Description

The `true` utility shall return with exit code zero.

4.65.3 Options

None.

4.65.4 Operands

None.

4.65.5 External Influences

4.65.5.1 Standard Input

None.

4.65.5.2 Input Files

None.

4.65.5.3 Environment Variables

None.

4.65.5.4 Asynchronous Events

Default.

4.65.6 External Effects

4.65.6.1 Standard Output

None.

4.65.6.2 Standard Error

None.

4.65.6.3 Output Files

None.

4.65.7 Extended Description

None.

4.65.8 Exit Status

The `true` utility always exits with a value of zero.

4.65.9 Consequences of Errors

Default.

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

Examples, Usage

The `true` utility is typically used in shell scripts. The special built-in utility `:` (see 3.14.2) is sometimes more efficient than `true`.

History of Decisions Made

The `true` utility has been retained in POSIX.2, even though the shell special built-in `:` provides similar functionality, because `true` is widely used in existing scripts and is less cryptic to novice human script readers.

4.66 `tty` — Return user's terminal name

4.66.1 Synopsis

`tty`

Obsolescent Version :

`tty –s`

4.66.2 Description

The `tty` utility shall write to the standard output the name of the terminal that is open as standard input. The name that is used shall be equivalent to the string that would be returned by the POSIX.1 {8} *ttyname* () function.

4.66.3 Options

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

The following option shall be supported by the implementation:

`–s` (Obsolescent.) Do not write the terminal name. Only the exit status shall be affected by this option. The terminal status shall be determined as if the POSIX.1 {8} *isatty* () function were used.

4.66.4 Operands

None.

4.66.5 External Influences

4.66.5.1 Standard Input

While no input is read from standard input, standard input shall be examined to determine whether or not it is a terminal, and/or to determine the name of the terminal.

4.66.5.2 Input Files

None.

4.66.5.3 Environment Variables

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

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 For the obsolescent version, 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.

4.66.5.4 Asynchronous Events

Default.

4.66.6 External Effects

4.66.6.1 Standard Output

If the `-s` option is specified, standard output shall not be used. If the `-s` option is not specified and standard input is a terminal device, a pathname of the terminal as specified by POSIX.1 {8} *ttyname* () shall be written in the following format:

`"%s\n"`, <*terminal name*>

Otherwise, a message shall be written indicating that standard input is not connected to a terminal. In the POSIX Locale, the `tty` utility shall use the format:

```
"not a tty\n"
```

4.66.6.2 Standard Error

Used only for diagnostic messages.

4.66.6.3 Output Files

None.

4.66.7 Extended Description

None.

4.66.8 Exit Status

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

- 0 Standard input is a terminal.
- 1 Standard input is not a terminal.
- >1 An error occurred.

4.66.9 Consequences of Errors

Default.

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

Examples, Usage

This utility checks the status of the file open as standard input against that of a system-defined set of files. It is possible that no match can be found, or that the match found need not be the same file as that which was opened for standard input (although they are the same device).

The `−s` option is useful only if the exit code is wanted. It does not rely on the ability to form a valid pathname. The `−s` option was made obsolescent because the same functionality is provided by `test -t 0`, but not dropped completely because historical scripts depend on this form.

History of Decisions Made

The definition of `tty` was made more explicit to explain the difference between a tty and a pathname of a tty.

4.67 `umask` — Get or set the file mode creation mask

4.67.1 Synopsis

`umask [−S] [`*mask*`]`

4.67.2 Description

The `umask` utility shall set the file mode creation mask of the current shell execution environment (see 3.12) to the value specified by the *mask* operand. This mask shall affect the initial value of the file permission bits of subsequently created files.

If the *mask* operand is not specified, the `umask` utility shall write to standard output the value of the invoking process's file mode creation mask.

4.67.3 Options

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

The following option shall be supported by the implementation:

`−S` Produce symbolic output.

The default output style is unspecified, but shall be recognized on a subsequent invocation of `umask` on the same system as a *mask* operand to restore the previous file mode creation mask.

4.67.4 Operands

The following operand shall be supported by the implementation:

mask A string specifying the new file mode creation mask. The string is treated in the same way as the *mode* operand described in 4.7.7 (`chmod` Extended Description).

For a *symbolic_mode* value, the new value of the file mode creation mask shall be the logical complement of the file permission bits portion of the file mode specified by the *symbolic_mode* string.

In a *symbolic_mode* value, the permissions *op* characters `+` and `-` shall be interpreted relative to the current file mode creation mask; `+` shall cause the bits for the indicated permissions to be cleared in the mask; `-` shall cause the bits for the indicated permissions to be set in the mask.

The interpretation of *mode* values that specify file mode bits other than the file permission bits is unspecified.

In the obsolescent octal integer form of *mode*, the specified bits shall be set in the file mode creation mask.

The file mode creation mask shall be set to the resulting numeric value.

As in `chmod`, application use of the octal number form for the *mode* values is obsolescent.

The default output of a prior invocation of `umask` on the same system with no operand shall also be recognized as a *mask* operand. The use of an operand obtained in this way is not obsolescent, even if it is an octal number.

4.67.5 External Influences

4.67.5.1 Standard Input

None.

4.67.5.2 Input Files

None.

4.67.5.3 Environment Variables

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

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.

4.67.5.4 Asynchronous Events

Default.

4.67.6 External Effects

4.67.6.1 Standard Output

When the *mask* operand is not specified, the `umask` utility shall write a message to standard output that can later be used as a `umask` *mask* operand.

If `−S` is specified, the message shall be in the following format:

> `"u=%s,g=%s,o=%s\n"`, *<owner permissions>* , *<group permissions>* , *<other permissions>*

where the three values shall be combinations of letters from the set {`r`, `w`, `x`}; the presence of a letter shall indicate that the corresponding bit is clear in the file mode creation mask.

If a *mask* operand is specified, there shall be no output written to standard output.

4.67.6.2 Standard Error

Used only for diagnostic messages.

4.67.6.3 Output Files

None.

4.67.7 Extended Description

None.

4.67.8 Exit Status

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

- 0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.
- >0 An error occurred.

4.67.9 Consequences of Errors

Default.

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

Examples, Usage

Since `umask` affects the current shell execution environment, it is generally pro-
vided as a shell regular built-in. If it is called in a subshell or separate utility 1
execution environment, such as one of the following: 1

```
(umask 002)                 1
nohup umask ...             1
find . -exec umask ... \;   1
```

it will not affect the file mode creation mask of the caller's environment. 1

The table mapping octal mode values in 4.7.7 does not require that the symbolic constants have those particular values.

In contrast to the negative permission logic provided by the file mode creation mask and the octal number form of the *mask* argument, the symbolic form of the *mask* argument specifies those permissions that are left alone.

Either of the commands:

```
umask a=rx,ug+w

umask 002
```

sets the mode mask so that subsequently created files have their S_IWOTH bit cleared.

After setting the mode mask with either of the above commands, the `umask` command can be used to write out the current value of the mode mask:

```
$ umask
0002
```

(The output format is unspecified, but historical implementations use the obsolescent octal integer mode format.)

```
$ umask -S
u=rwx,g=rwx,o=rx
```

Either of these outputs can be used as the mask operand to a subsequent invocation of the `umask` utility.

Assuming the mode mask is set as above, the command:

```
umask g-w
```

sets the mode mask so that subsequently created files have their S_IWGRP , and S_IWOTH bits cleared.