A related application of the `ss` request is to insert discardable horizontal space; i.e., space that is discarded at a line break. For example, some footnote styles collect the notes into a single paragraph with large spaces between each.

```
.ie n .ll 50n
.el   .ll 2.75i
.ss 12 48
1. J. Fict. Ch. Soc. 6 (2020), 3\[en]14.
2. Better known for other work.
```

The result has obvious inter-sentence spacing.

```
1. J. Fict. Ch. Soc. 6 (2020), 3-14.     2. Better
known for other work.
```

If *undiscardable* space is required, use the `\h` escape.

## 5.8 Manipulating Hyphenation

GNU `troff` normally hyphenates words where necessary. The machine-driven determination of hyphenation points in words requires algorithms and data, and is susceptible to conventions and preferences. Before tackling such *automatic hyphenation*, let us consider how hyphenation points can be set manually.

Explicitly hyphenated words such as "mother-in-law" are eligible for breaking after each of their hyphens when GNU `troff` fills lines. Relatively few words in a language offer such obvious break points, however, and automatic hyphenation is not perfect, particularly for unusual words found in technical literature. We may wish to instruct GNU `troff` how to hyphenate specific words if the need arises.

`.hw` *word ...* [Request]

Define each *hyphenation exception word* with each hyphen '-' in the word indicating a hyphenation point. For example, the request

```
.hw in-sa-lub-rious alpha
```

marks potential hyphenation points in "insalubrious", and prevents "alpha" from being hyphenated at all.

Besides the space character, any character whose hyphenation code is zero can be used to separate the arguments of `hw` (see the `hcode` request below). In addition, this request can be used more than once.

Hyphenation points specified with `hw` are not subject to the within-word placement restrictions imposed by the `hy` request (see below).

Hyphenation exceptions specified with the `hw` request are associated with the hyphenation language (see below) and environment (see Section 5.26 [Environments], page 179); calling the `hw` request in the absence of a hyphenation language is an error.

The request is ignored if there are no parameters.

These are known as hyphenation *exceptions* in the expectation that most users will avail themselves of automatic hyphenation; these exceptions override any rules that would normally apply to a word matching a hyphenation exception defined with `hw`.

Situations also arise when only a specific occurrence of a word needs its hyphenation altered or suppressed, or when something that is not a word in a natural language, like a URL, needs to be broken in sensible places without hyphens.

`\%` [Escape]
`\:` [Escape]

To tell GNU `troff` how to hyphenate words as they occur in input, use the `\%` escape, also known as the *hyphenation character*. Each instance within a word indicates to GNU `troff` that the word may be hyphenated at that point, while prefixing a word with this escape prevents it from being otherwise hyphenated. This mechanism affects only that occurrence of the word; to change the hyphenation of a word for the remainder of the document, use the `hw` request.

GNU `troff` regards the escapes `\X` and `\Y` as starting a word; that is, the `\%` escape in, say, '`\X'...'\%foobar`' or '`\Y'...'\%foobar`' no longer prevents hyphenation of '`foobar`' but inserts a hyphenation point just prior to it; most likely this isn't what you want. See Section 5.30 [Post-processor Access], page 188.

The `\:` escape inserts a non-printing break point; that is, the word can break there, but the soft hyphen glyph is not written to the output if it does. This escape is an input word boundary, so the remainder of the word is subject to hyphenation as normal.

You can use `\:` and `\%` in combination to control breaking of a file name or URL or to permit hyphenation after only certain explicit hyphens within a word.

```
The \%Lethbridge-Stewart-\:\%Sackville-Baggins divorce
was, in retrospect, inevitable once the contents of
\%/var/log/\:\%httpd/\:\%access_log on the family web
server came to light, revealing visitors from Hogwarts.
```

`.hc` [*char*] [Request]

Change the hyphenation character to *char*. This character then works as the `\%` escape normally does, and thus no longer appears in the output.[23] Without an argument, `hc` resets the hyphenation character to `\%` (the default).

The hyphenation character is associated with the current environment (see Section 5.26 [Environments], page 179).

[23] `\%` itself stops marking hyphenation points but still produces no output glyph.

`.shc` [*glyph*] [Request]

Set the *soft hyphen character*,[24] inserted when a word is hyphenated automatically or at a hyphenation character `\%`, to *glyph*.[25] If the argument is omitted, the soft hyphen glyph is set to the default, `\[hy]`. If the selected glyph does not exist in the font in use at a potential hyphenation point, then the line is not broken at that point. Neither character definitions (specified with the `char` and similar requests) nor translations (specified with the `tr` request) are considered when assigning the soft hyphen glyph.

Several requests influence automatic hyphenation. Because conventions vary, a variety of hyphenation modes is available to the `hy` request; these determine whether hyphenation will apply to a word prior to breaking a line at the end of a page (more or less; see below for details), and at which positions within that word automatically determined hyphenation points are permissible. The places within a word that are eligible for hyphenation are determined by language-specific data and lettercase relationships. Furthermore, hyphenation of a word might be suppressed because too many previous lines have been hyphenated (`hlm`), the line has not reached a certain minimum length (`hym`), or the line can instead be adjusted with up to a certain amount of additional inter-word space (`hys`).

`.hy` [*mode*] [Request]
`\n[.hy]` [Register]

Set automatic hyphenation mode to *mode*. The optional numeric argument *mode* encodes conditions for hyphenation.

Typesetting practice generally does not avail itself of every opportunity for hyphenation, but the details differ by language and site mandates. The hyphenation modes of AT&T `troff` were implemented with English-language publishing practices of the 1970s in mind, not a scrupulous enumeration of conceivable parameters. GNU `troff` extends those modes such that finer-grained control is possible, favoring compatibility with older implementations over a more intuitive arrangement. The means of hyphenation mode control is a set of numbers that can be added up to encode the behavior sought.[26] The entries in the table below are termed *values*, and the sum of the desired values is the *mode*.

| | |
|---|---|
| `0` | disables hyphenation. |
| `1` | enables hyphenation except after the first and before the last character of a word; this is the default if *mode* is omitted and also the start-up value of GNU `troff`. |

[24] "Soft hyphen *character*" is a misnomer since it is an output glyph.

[25] It is "soft" because it only appears in output where hyphenation is actually performed; a "hard" hyphen, as in "long-term", always appears.

[26] The mode is a vector of booleans encoded as an integer. To a programmer, this fact is easily deduced from the exclusive use of powers of two for the configuration parameters; they are computationally easy to "mask off" and compare to zero. To almost everyone else, the arrangement seems recondite and unfriendly.

The remaining values "imply" 1; that is, they enable hyphenation under the same conditions as '`.hy 1`', and then apply or lift restrictions relative to that basis.

| | |
|---|---|
| `2` | disables hyphenation of the last word on a page.[27] |
| `4` | disables hyphenation before the last two characters of a word. |
| `8` | disables hyphenation after the first two characters of a word. |
| `16` | enables hyphenation before the last character of a word. |
| `32` | enables hyphenation after the first character of a word. |

Apart from value 2, restrictions imposed by the hyphenation mode are *not* respected for words whose hyphenations have been explicitly specified with the hyphenation character ('`\%`' by default) or the `hw` request.

The nonzero values in the previous table are additive. For example, value 12 causes GNU `troff` to hyphenate neither the last two nor the first two characters of a word. Some values cannot be used together because they contradict; for instance, values 4 and 16, and values 8 and 32. As noted, it is superfluous to add 1 to any nonzero even mode.

The automatic placement of hyphens in words is determined by *pattern files*, which are derived from TeX and available for several languages. The number of characters at the beginning of a word after which the first hyphenation point should be inserted is determined by the patterns themselves; it can't be reduced further without introducing additional, invalid hyphenation points (unfortunately, this information is not part of a pattern file—you have to know it in advance). The same is true for the number of characters at the end of a word before the last hyphenation point should be inserted. For example, you can supply the following input to '`echo $(nroff)`'.

```
.ll 1
.hy 48
splitting
```

You will get

```
s- plit- t- in- g
```

instead of the correct 'split- ting'. U.S. English patterns as distributed with GNU `troff` need two characters at the beginning and three characters at the end; this means that value 4 of `hy` is mandatory. Value 8 is possible as an additional restriction, but values 16 and 32 should be avoided, as should mode 1 (the default!). Modes 4 and 6 are typical.

[27] This value prevents hyphenation if the next page location trap is closer than the next text baseline would be. GNU `troff` automatically inserts an implicit vertical position trap at the end of each page to cause a page transition. Users or macro packages can set such traps explicitly to prevent hyphenation of the last word in a column in multi-column page layouts or before floating figures or tables. See Section 5.24.1.1 [Page Location Traps], page 167.

A table of left and right minimum character counts for hyphenation as needed by the patterns distributed with GNU `troff` follows; see the *groff_tmac*(5) man page for more information on GNU `troff`'s language macro files.

| **language** | **pattern name** | **left min** | **right min** |
|---|---|---|---|
| Czech | cs | 2 | 2 |
| U.S. English | us | 2 | 3 |
| French | fr | 2 | 3 |
| German traditional | det | 2 | 2 |
| German reformed | den | 2 | 2 |
| Swedish | sv | 1 | 2 |

Hyphenation exceptions within pattern files (i.e., the words within a TEX `\hyphenation` group) also obey the hyphenation restrictions given by `hy`. However, exceptions specified with `hw` do not.

The hyphenation mode is associated with the current environment (see Section 5.26 [Environments], page 179).

The hyphenation mode can be found in the read-only register '`.hy`'.

`.nh` [Request]

Disable automatic hyphenation; i.e., set the hyphenation mode to 0 (see above). The hyphenation mode of the last call to `hy` is not remembered.

`.hpf` *pattern-file* [Request]
`.hpfa` *pattern-file* [Request]
`.hpfcode` *a b* [*c d*] ... [Request]

Read hyphenation patterns from *pattern-file*. This file is sought in the same way that macro files are with the `mso` request or the `-m`*`name`* command-line option to `groff`.

The *pattern-file* should have the same format as (simple) TEX pattern files. More specifically, the following scanning rules are implemented.

- A percent sign starts a comment (up to the end of the line) even if preceded by a backslash.
- "Digraphs" like `\$` are not supported.
- `^^`*`xx`* (where each *x* is 0–9 or a–f) and `^^`*`c`* (character *c* in the code point range 0–127 decimal) are recognized; other uses of `^` cause an error.
- No macro expansion is performed.
- `hpf` checks for the expression `\patterns{...}` (possibly with white-space before or after the braces). Everything between the braces is taken as hyphenation patterns. Consequently, `{` and `}` are not allowed in patterns.
- Similarly, `\hyphenation{...}` gives a list of hyphenation exceptions.
- `\endinput` is recognized also.

- For backwards compatibility, if `\patterns` is missing, the whole file is treated as a list of hyphenation patterns (except that the `%` character is recognized as the start of a comment).

The `hpfa` request appends a file of patterns to the current list.

The `hpfcode` request defines mapping values for character codes in pattern files. It is an older mechanism no longer used by GNU `troff`'s own macro files; for its successor, see `hcode` below. `hpf` or `hpfa` apply the mapping after reading the patterns but before replacing or appending to the active list of patterns. Its arguments are pairs of character codes—integers from 0 to 255. The request maps character code *a* to code *b*, code *c* to code *d*, and so on. Character codes that would otherwise be invalid in GNU `troff` can be used. By default, every code maps to itself except those for letters 'A' to 'Z', which map to those for 'a' to 'z'.

The set of hyphenation patterns is associated with the language set by the `hla` request. The `hpf` request is usually invoked by the `troffrc` or `troffrc-end` file; by default, `troffrc` loads hyphenation patterns and exceptions for U.S. English (in files `hyphen.us` and `hyphenex.us`).

A second call to `hpf` (for the same language) replaces the hyphenation patterns with the new ones.

Invoking `hpf` or `hpfa` causes an error if there is no hyphenation language.

If no `hpf` request is specified (either in the document, in a `troffrc` or `troffrc-end` file, or in a macro package), GNU `troff` won't automatically hyphenate at all.

`.hcode` *c1 code1* [*c2 code2*] . . . [Request]

Set the hyphenation code of character *c1* to *code1*, that of *c2* to *code2*, and so on. A hyphenation code must be a single input character (not a special character) other than a digit or a space. The request is ignored if it has no parameters.

For hyphenation to work, hyphenation codes must be set up. At start-up, GNU `troff` assigns hyphenation codes to the letters '`a`'–'`z`' (mapped to themselves), to the letters '`A`'–'`Z`' (mapped to '`a`'–'`z`'), and zero to all other characters. Normally, hyphenation patterns contain only lowercase letters which should be applied regardless of case. In other words, they assume that the words 'FOO' and 'Foo' should be hyphenated exactly as 'foo' is. The `hcode` request extends this principle to letters outside the Unicode basic Latin alphabet; without it, words containing such letters won't be hyphenated properly even if the corresponding hyphenation patterns contain them. For example, the following `hcode` requests are necessary to assign hyphenation codes to the letters '`ÄäÖöÜüß`' (needed for German):

```
.hcode ä ä  Ä ä
.hcode ö ö  Ö ö
.hcode ü ü  Ü ü
.hcode ß ß
```

Without those assignments, GNU `troff` treats German words like ‘Kindergärten’ (the plural form of ‘kindergarten’) as two substrings ‘kinderg’ and ‘rten’ because the hyphenation code of the umlaut a is zero by default. There is a German hyphenation pattern that covers ‘kinder’, so GNU `troff` finds the hyphenation ‘kin-der’. The other two hyphenation points (‘kin-der-gär-ten’) are missed.

`.hla` *lang* [Request]
`\n[.hla]` [Register]

Set the hyphenation language to *lang*. Hyphenation exceptions specified with the `hw` request and hyphenation patterns and exceptions specified with the `hpf` and `hpfa` requests are associated with the hyphenation language. The `hla` request is usually invoked by the `troffrc` or `troffrc-end` files; `troffrc` sets the default language to ‘`us`’ (U.S. English).

The hyphenation language is associated with the current environment (see Section 5.26 [Environments], page 179).

The hyphenation language is available as a string in the read-only register ‘`.hla`’.

```
    .ds curr_language \n[.hla]
    \*[curr_language]
        ⇒ us
```

`.hlm` [*n*] [Request]
`\n[.hlm]` [Register]
`\n[.hlc]` [Register]

Set the maximum number of consecutive hyphenated lines to *n*. If *n* is negative, there is no maximum. If omitted, *n* is $-1$. This value is associated with the current environment (see Section 5.26 [Environments], page 179). Only lines output from a given environment count towards the maximum associated with that environment. Hyphens resulting from `\%` are counted; explicit hyphens are not.

The `.hlm` read-only register stores this maximum. The count of immediately preceding consecutive hyphenated lines is available in the read-only register `.hlc`.

`.hym` [*length*] [Request]
`\n[.hym]` [Register]

Set the (right) hyphenation margin to *length*. If the adjustment mode is not ‘`b`’ or ‘`n`’, the line is not hyphenated if it is shorter than *length*. Without an argument, the hyphenation margin is reset to its default value, 0. The default scaling indicator is ‘`m`’. The hyphenation margin is associated with the current environment (see Section 5.26 [Environments], page 179).

A negative argument resets the hyphenation margin to zero, emitting a warning of type ‘`range`’.

The hyphenation margin is available in the `.hym` read-only register.

`.hys` [*hyphenation-space*] [Request]
`\n[.hys]` [Register]

Suppress hyphenation of the line in adjustment modes '`b`' or '`n`' if it can be justified by adding no more than *hyphenation-space* extra space to each inter-word space. Without an argument, the hyphenation space adjustment threshold is set to its default value, 0. The default scaling indicator is '`m`'. The hyphenation space adjustment threshold is associated with the current environment (see Section 5.26 [Environments], page 179).

A negative argument resets the hyphenation space adjustment threshold to zero, emitting a warning of type '`range`'.

The hyphenation space adjustment threshold is available in the `.hys` read-only register.

## 5.9 Manipulating Spacing

`.sp` [*distance*] [Request]

Space downwards *distance*. With no argument it advances 1 line. A negative argument causes `gtroff` to move up the page the specified distance. If the argument is preceded by a '`|`' then `gtroff` moves that distance from the top of the page. This request causes a line break, and that adds the current line spacing to the space you have just specified. The default scaling indicator is '`v`'.

For convenience you may wish to use the following macros to set the height of the next line at a given distance from the top or the bottom of the page:

```
.de y-from-top-down
.  sp |\\$1-\\n[.v]u
..
.
.de y-from-bot-up
.  sp |\\n[.p]u-\\$1-\\n[.v]u
..
```

A call to '`.y-from-bot-up 10c`' means that the bottom of the next line will be at 10 cm from the paper edge at the bottom.

If a vertical trap is sprung during execution of `sp`, the amount of vertical space after the trap is discarded. For example, this