GNUS Manual

Fourth Edition, GNUS Version 3.14

March 1990

Masanobu UMEDA

Distribution

GNUS is *free*; this means that everyone is free to use it and free to redistribute it on a free basis. GNUS is not in the public domain; it is copyrighted and there are restrictions on its distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of GNUS that they might get from you. The precise conditions appears following this section.

The easiest way to get a copy of GNUS is from someone else who has it. You need not ask for permission to do so, or tell any one else; just copy it.

If you have access to the Internet, you can get the latest version of GNUS from tut.cis.ohio-state.edu [128.146.8.60] using anonymous ftp. You can also get it from osu-cis using anonymous uucp.

You may also receive GNUS when you buy a computer. Computer manufacturers are free to distribute copies on the same terms that apply to everyone else. These terms require them to give you the full sources, including whatever changes they may have made, and to permit you to redistribute the GNUS received from them under the usual terms of the General Public License. In other words, the program must be free for you when you get it, not just free for the manufacturer.

If you cannot get a copy in any of those ways, you can order one from the author using electronic mail. For further information, write to:

umerin@mse.kyutech.ac.jp

If you cannot reach the author using these addresses, try the following mailing list:

info-gnus-english@tut.cis.ohio-state.edu

The list is intended to exchange useful information about GNUS, such as bug reports, useful hooks, and extensions. Feel free to ask about the nearest machine that has GNUS installed. Subscription requests for this list should be sent to:

info-gnus-english-request@tut.cis.ohio-state.edu

See Appendix C [Reporting Bugs], page 61, for more information on the mailing list.

GNUS General Public License

(Clarified 13 Jan 1989)

The license agreements of most software companies keep you at the mercy of those companies. By contrast, our general public license is intended to give everyone the right to share GNUS. To make sure that you get the rights we want you to have, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. Hence this license agreement.

Specifically, we want to make sure that you have the right to give away copies of GNUS, that you receive source code or else can get it if you want it, that you can change GNUS or use pieces of it in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of GNUS, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for GNUS. If GNUS is modified by someone else and passed on, we want its recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

Therefore I (Masanobu Umeda) use the following terms, which are written for GNU Emacs by Richard Stallman and the Free Software Foundation, Inc. They say what you must do to be allowed to distribute or change GNU Emacs. You can know your right to distribute or change GNUS by replacing any occurrences of GNU Emacs with GNUS. Please apply the same policies to GNUS as GNU Emacs.

Copying Policies

1. You may copy and distribute verbatim copies of GNU Emacs source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each file a valid copyright notice "Copyright © 1988 Free Software Foundation, Inc." (or with whatever year is appropriate); keep intact the notices on all files that refer to this License Agreement and to the absence of any warranty; and give any other recipients of the GNU Emacs program a copy of this License Agreement along with the program. You may charge a distribution fee for the physical act of transferring a copy.
2. You may modify your copy or copies of GNU Emacs source code or any portion of it, and copy and distribute such modifications under the terms of Paragraph 1 above, provided that you also do the following:
   - cause the modified files to carry prominent notices stating who last changed such files and the date of any change; and
   - cause the whole of any work that you distribute or publish, that in whole or in part contains or is a derivative of GNU Emacs or any part thereof, to be licensed at no charge to all third parties on terms identical to those contained in this License Agreement (except that you may choose to grant more extensive warranty protection to some or all third parties, at your option).
   - if the modified program serves as a text editor, cause it, when started running in the simplest and usual way, to print an announcement including a valid copyright

Acknowledgments

The author would like to thank all those who have helped design and develop GNUS, and who tested non stable code at their own risk.

Haruo Yokota, Hiroyuki Yoshida, and Kohji Akiyama contributed many valuable suggestions in the early stages of the development of GNUS. Nobuyuki Hiraoka, Tsuneo Nakata, and Mark Feldman patiently tested non stable code and contributed valuable comments. Mark Feldman also helped with proofreading. Mike Khaw, Kouich Kumon, and Mitsuhiro Kishimoto helped contribute the name GNUS. Itaru Ichikawa helped implement the caesar rotations for Japanese characters. Yasunari Itoh, Akio Shibata, and Ken-ichi Sakaeda made this software available on SX/A Emacs and some old versions of GNU Emacs. Leonard H. Tower Jr., Bob Sutterfield, and Karl Kleinpaste helped create mailing lists for GNUS users. Bob Sutterfield has also been managing the info-gnus-english mailing list. Jordan K. Hubbard helped with proofreading of this Texinfo manual.

The author would also like to thank many other GNUS users who have sent comments, suggestions, and bug fixes. Thanks also go to Richard Stallman and the Free Software Foundation for their good work.

Introduction

GNUS is a program for reading and writing USENET news using GNU Emacs. This manual documents the use and simple customization of GNUS.

Unlike other conventional newsreaders such as rn which was (:-) the most popular newsreader in the world, GNUS runs inside the GNU Emacs editor as a subsystem. This means that there is no need to invoke an editor when composing articles or mail. The reading and writing of articles can be done in the same Emacs environment you usually work in.

Like rrn, a remote version of rn, GNUS can use computer networks for retrieving articles. This means that there is no need to have a local copy of the news spool, to mount a remote file system over the network, or to run Emacs on a remote machine. Its great advantage is to balance loads and exploit resources of the entire computer system in a distributed environment. The protocol used by GNUS is *NNTP*, the Network News Transfer Protocol, defined by RFC977.

Unlike rrn, GNUS can talk to many NNTP servers easily. The only thing you have to do is to create startup files for each NNTP server.

Like other libraries of GNU Emacs, GNUS is completely written in Emacs lisp. This means that GNUS is highly extensible and customizable just like GNU Emacs. It is possible to change the behavior of GNUS and extend its functions by using variables and hooks.

GNUS is pronounced **“NUZ”**. Never call it **“ghu-NUZ”** nor **“ghu-NAS”**.

This manual is currently in progress and thus is incomplete. Your suggestions, bug fixes, and contributions are welcome.

1 Installing GNUS

Installation of GNUS and some initialization of your computing environment are described in this chapter. Please read the following sections carefully before getting started with GNUS.

1.1 Files of GNUS

Unpacking the shar files will produce the following files. They are the Emacs lisp sources, a C source, a Texinfo manual of GNUS, and a Makefile.

`gnus.el`	Main part of GNUS newsreader.
`gnuspost.el`	Post news commands.
`gnusmail.el`	Mail reply commands.
`gnusmisc.el`	Miscellaneous commands.
`nntp.el`	NNTP package.
`nnspool.el`	A package accessing local news spool like NNTP.
`mhspool.el`	A package accessing private directory like NNTP.
`timezone.el`	A general time zone handling package useful not only for GNUS.
`tcp.el`	Patches to some versions of GNU Emacs which do not have the function `open-network-stream`.
`tcp.c`	C program for external TCP/IP implementation. This is used with `tcp.el`.
`gnus.texinfo`	Texinfo manual of GNUS.
`Makefile`	Makefile to byte-compile the lisp files using the `make` command.

1.2 Byte-Compilation

Move the lisp files, the C file, and `Makefile` to the appropriate directory in the search path defined by the variable `load-path`. Before actually byte-compiling the lisp files, make sure there are no byte-compiled files of older versions of GNUS in that directory. Remove or rename such files as the byte-compiler may be confused by old macro definitions. If you can use the `make` command, you don't have to take care of the dependencies.

The C file `tcp.c` should be compiled with a C compiler and installed in a directory in the search path defined by the variable `exec-path`, if this is required.

If you can use the `make` command, just type `make` in a Unix shell. All the lisp files will be byte-compiled. Otherwise, byte-compile lisp files in the following order according to your computing environment by yourself:

1. Byte-compile `nntp.el`, `gnus.el`, `gnuspost.el`, `gnusmail.el`, and `gnusmisc.el` in this order.
2. Byte-compile `timezone.el` if you didn't have it.
3. Byte-compile `nnspool.el` if you want to use the local news spool of your machine instead of NNTP (see Section 2.2 [Local News Spool], page 17).
4. Byte-compile `mhspool.el` if you want to read articles or mail in your private directory using GNUS (see Section 2.3 [Private Directory], page 17).
5. Compile and install `tcp.el` and `tcp.c` if TCP/IP is not supported by Emacs but is supported by your operating system.

`tcp.el` defines the function `open-network-stream`, and `tcp.c` is an emulation program for the stream used by the function. If you modified `tcp.c` for your system, please send the author the diffs. Some of them will be included in the future releases of GNUS.

1.3 Autoloading

It is useful to define autoload entries in `.emacs`, `site-init.el` or `default.el` as follows:

```
(autoload 'gnus "gnus" "Read network news." t)
(autoload 'gnus-post-news "gnuspost" "Post a new news." t)
```

1.4 Environment

The NNTP server and its service name, your domain and organization, and other important definitions of your computing environment are described in this section. Since these definitions depend heavily on your environment, you'd better be familiar with the operating system you are using. Knowledge of the USENET software is also important.

1.4.1 NNTP Server

The variable `gnus-nntp-server` specifies the default NNTP server. To define the server '`flab`', put the following code in `.emacs`, `site-init.el` or `default.el`:

```
(setq gnus-nntp-server "flab")
```

The variable `gnus-nntp-server` is initialized from the `NNTPSERVER` environment variable. To define the server using the `NNTPSERVER` environment variable, put the following code in `.login`:

```
setenv NNTPSERVER "flab"
```

If an NNTP server is preceded by a colon such as '`:Mail`', the user's private directory `~/Mail` is used as the news spool. This makes it possible to read mail stored in MH folders or articles saved by GNUS. See Section 2.3 [Private Directory], page 17, for more information.

GNUS will ask you for the NNTP server at start up time unless it is defined. Even if the default server is defined, it is possible to choose another server by invoking GNUS with a prefix argument like *`C-u M-x gnus`* (see Section 2.1 [Getting Started], page 17).

1.4.2 NNTP Service

The default service name of NNTP is `"nntp"`. You may, however, have to define the service name as the number `119` as follows:

```
(setq gnus-nntp-service 119)
```

If you'd like to use a local news spool of your machine directly instead of NNTP, set the variable to `nil` as follows:

```
(setq gnus-nntp-service nil)
```

In this case, the NNTP server must be a local host name returned by the function `system-name` (see Section 2.2 [Local News Spool], page 17).

1.4.3 Domain and Organization

Domain and *organization* must be defined before you post your first article, because they are included in all articles you post and will be used for identifying who you are.

Domain is the domain part of your mail address excluding the local host name. For example, if your mail address is '`umerin@photon.stars.flab.Fujitsu.CO.JP`' and the local host name is '`photon`', your domain is '`stars.flab.Fujitsu.CO.JP`'. If the function `system-name` of your Emacs returns the full Internet name, you do not have to define the domain.

Organization is the organization you belong to. It must be defined unless it is defined in the file `/usr/lib/news/organization`.

To define the domain '`stars.flab.Fujitsu.CO.JP`' and the organization '`Fujitsu Laboratories Ltd., Kawasaki, Japan.`' using lisp variables, put the following code in `.emacs`, `site-init.el` or `default.el`. If you are a system administrator and are installing GNUS for other users, `site-init.el` is the best place to define this because the domain and organization are common to all users of the system.

```
(setq gnus-local-domain "stars.flab.Fujitsu.CO.JP")
(setq gnus-local-organization
      "Fujitsu Laboratories Ltd., Kawasaki, Japan.")
```

The `DOMAINNAME` and `ORGANIZATION` environment variables are used instead, if defined. To define these variables, put the following code in `.login`.

```
setenv DOMAINNAME   "stars.flab.Fujitsu.CO.JP"
setenv ORGANIZATION "Fujitsu Laboratories Ltd., Kawasaki, Japan."
```

If the value of the `ORGANIZATION` environment variable or the variable `gnus-local-organization` begins with a slash, it is taken as the name of a file whose contents are read for the value. If neither of these is defined, and a file `~/.organization-`*`distribution`* or `~/.organization` exists, the contents of that file are used. If neither of them does not exist, and the file `/usr/lib/news/organization` exists, its contents are used.

1.4.4 GENERICFROM

If the variable `gnus-use-generic-from` is non-`nil`, the local host name of your machine will not appear in the '`From:`' field of article headers you post. This is called the *GENERICFROM* feature in the Bnews system. This may be useful if there are many workstations connected to each other in a local area network, and aliases service or automatic forwarding of mail is supported between the workstations.

To use the GENERICFROM, put the following code in `.emacs`, `site-init.el` or `default.el`. If you are a system administrator and are installing GNUS for other users, `site-init.el` is the best place to define it because the definition is common to all users of the system having the same domain and organization (see Section 1.4.3 [Domain and Organization], page 13).

```
(setq gnus-use-generic-from t)
```

As a special case of the GENERICFROM feature, if the variable `gnus-use-generic-from` is a string, it is used as your domain instead of the definition of the environment variable `DOMAINNAME` or the variable `gnus-local-domain` (see Section 1.4.3 [Domain and Organization], page 13).

1.4.5 GENERICPATH

If the variable `gnus-use-generic-path` is `nil`, the NNTP server name followed by the user login name is used in the '`Path:`' field of article headers you post. If it is a string, the string followed by the user login name is used instead. Otherwise, if it is non-`nil`, only the user login name is used. This is called the *GENERICPATH* feature in the Bnews system.

For example, to define the generic path '`flab`', put the following codes in `.emacs`, `site-init.el` or `default.el`. If you are a system administrator and are installing GNUS for other users, `site-init.el` is the best place to define it because the definition is common to all users of the system having the same domain and organization (see Section 1.4.3 [Domain and Organization], page 13).

```
(setq gnus-use-generic-path "flab")
```

In this case, the '`Path:`' field will be generated as '`Path: flab!`*`user`*'.

1.4.6 Startup File

Startup file is a file recording information on articles you have already read. GNUS uses `.newsrc` for the startup file as in the Bnews system. If you think you will talk to exactly one NNTP server, you can use it without any problems. Otherwise, if you want to talk to several NNTP servers, you'd better use server specific startup files since startup files are not portable between servers. The server specific startup file for an NNTP server on a machine *server* is a file named `.newsrc-`*`server`*. For example, `.newsrc-photon` is for an NNTP server on a machine named '`photon`'. The primary name of the startup file, `.newsrc`, is specified by the variable `gnus-startup-file` (see Section A.1 [Variables], page 41).

GNUS automatically adds newly created newsgroups to a startup file when getting started. To prevent adding the newsgroups under some newsgroup hierarchies, you can use the options line in the startup file or the variable `gnus-subscribe-newsgroup-method` provided for subscription method customization. See Section A.1 [Variables], page 41, for more information on the variable customization.

Option `-n` of the options line in the startup file is recognized properly the same as for the Bnews system. For example, if the options line is '`options -n !talk talk.rumors`', newsgroups under the '`talk`' hierarchy except for '`talk.rumors`' are ignored while checking new newsgroups. These ignored newsgroups can be added manually using the command *`U`* (`gnus-Group-unsubscribe-group`) in the Newsgroup buffer. Use the commands *`C-k`* and *`C-w`* (`gnus-Group-kill-group` and `gnus-Group-kill-region`) to kill newsgroups from the startup file per a newsgroup basis. See Section 4.3 [Maintenance], page 22, for more information.

Once a startup file is updated by GNUS, the *quick startup file* of which the file name is generated by appending `.el` to that of the raw startup file is also created. The quick startup file can be read by Emacs faster than the raw startup file since all information in the file is in lisp form. If there is a quick startup file and it is newer than the raw startup file, the quick startup file is loaded instead of the raw startup file. If the raw startup file is newer, it is normally read after loading the quick startup file. You should not remove the quick startup file because it contains additional information. Instead, make the raw startup file newer than that by touching it or force GNUS to read it by using the command *R* (`gnus-Group-restart`) in the Newsgroup buffer if you want to reflect the changes of the raw startup file to GNUS.

1.5 Texinfo Manual

`gnus.texinfo` is a manual of GNUS written in Texinfo format. This file can be printed using $\TeX$, and also can be read using Info Mode of Emacs.

See Info file `texinfo`, node '`Creating an Info File`', to create an on-line Info file from the Texinfo manual. If you are not allowed to create the Info file in the standard Info directory specified by the variable `Info-directory`, create it in your private directory and set the variable `gnus-Info-directory` to the directory.

If this Info file is installed, you can read the documentation of GNUS according to the current major mode of GNUS. The command `gnus-Info-find-node` for reading appropriate Info nodes of the Info file is assigned to *C-c C-i* in all major modes of GNUS.

See Info file `texinfo`, node '`Printing Hardcopy`', to print a hardcopy of the manual.

2 Starting up GNUS

2.1 Getting Started GNUS

Type *M-x gnus* at the top level of GNU Emacs to invoke GNUS. Unless a default NNTP server is defined, you will be asked for the name of the server. The NNTP server can be changed at startup time by giving a prefix argument to the command even if a default server is defined. See Section 1.4.1 [NNTP Server], page 12, to define the default server.

M-x gnus Run GNUS using the default NNTP server.

C-u M-x gnus
Run GNUS without using the default NNTP server.

2.2 Using Local News Spool

If the NNTP server is the local host name (exactly the same as the value returned by the function `system-name`), and the variable `gnus-nntp-service` is nil, GNUS looks up the local news spool of your machine directly instead of NNTP (see Section 1.4.2 [NNTP Service], page 13).

It may be a good idea not to use NNTP if you wish to reduce network traffic (even if an NNTP server is running on the local machine) though commands for retrieving articles by Message-IDs may take longer or not work at all. See Section 5.3 [Referencing Articles], page 30, to refer to articles by the Message-IDs, and see Section A.3 [Spool Variables], page 49, for more information on the restrictions.

2.3 Reading a Private Directory

If an NNTP server is preceded by a colon such as '`:Mail`', the user's private directory `~/Mail` is used as the news spool. This makes it possible to read mail stored in MH folders or articles saved by GNUS. An active file for the directory is generated from files of which the file name consists of only numeric characters, and other files containing non-numeric characters in their file name are ignored.

A server specific startup file for each directory must be created before starting GNUS. For example, a startup file for the directory `~/Mail` is a file named `.newsrc-:Mail`. See Section 1.4.6 [Startup File], page 14, for more information on the server specific startup file.

3 Buffers of GNUS

Basically GNUS uses three buffers in Emacs: Newsgroup buffer, Subject buffer, and Article buffer. Each of them is associated with a specific major mode of Emacs. The configuration of the windows displaying these buffers are customizable by using the variable `gnus-window-configuration`. See Section A.1 [Variables], page 41, for more information on customizing the window configuration.

3.1 Newsgroup Buffer

Newsgroup buffer is for listing newsgroups. The major mode of this buffer is *Group Mode*. The buffer is created and popped up when GNUS starts. Newsgroups which are subscribed to and contain unread articles are usually listed in the buffer. If there is no such newsgroup, the buffer will be empty and a message '`No news is good news.`' will be displayed in the echo area of Emacs.

The contents of a Newsgroup buffer looks something like the following:

```
SM  NUMBER: NEWSGROUP
```

S
: A character indicating whether the newsgroup is subscribed to. '`U`' means the newsgroup is unsubscribed. '` `' means it is subscribed to.

M
: A character indicating whether there are unread articles in the newsgroup. '`*`' means there are no newly arrived articles in the newsgroup. '` `' means there are newly arrived articles.

NUMBER
: The number of unread articles in the newsgroup.

NEWSGROUP
: The name of the newsgroup.

3.2 Subject Buffer

Subject buffer is for listing subjects and other important headers of articles. The major mode of the buffer is *Subject Mode*. The buffer is created for each newsgroup when the newsgroup is selected in the Newsgroup buffer.

The contents of a Subject buffer looks something like the following:

```
S NUMBER:C[LINES:AUTHOR] SUBJECT
```

S
: A character indicating whether an article is newly arrived. '` `' means the article is newly arrived. '`D`' means it was read and marked so. '`-`' means it was read once but marked as unread again.

NUMBER
: The number assigned to the article.

C
: A character indicating which article is currently selected. '`+`' is placed on the current article.

LINES
: The number of lines of the article body.

AUTHOR
: Mail address of the author of the article.

SUBJECT
: Subject of the article.

Strings between '`[`' and '`]`' are optional, and can be customized using the variable **`gnus-optional-headers`**. See Section A.1 [Variables], page 41, and see Section A.5 [Hooks], page 50, for more information on customization.

3.3 Article Buffer

Article buffer is for displaying articles. The major mode of the buffer is *Article Mode*. Very few commands are available in this buffer because most operations on the buffer can be done in the Subject buffer.

4 Newsgroup Commands

The Newsgroup buffer is intended to show newsgroups which are subscribed to and contain unread articles. Unsubscribed newsgroups or newsgroups containing no unread articles are not usually displayed in the buffer.

Commands for browsing through newsgroups, selecting newsgroups to read, and maintaining newsgroups are described in this chapter.

4.1 Walking Around Newsgroups

First of all, you have to move the point to a newsgroup to read articles in it.

`n`
: Move point to the next newsgroup containing unread articles (`gnus-Group-next-unread-group`).

`p`
`DEL`
: Move point to the previous newsgroup containing unread articles (`gnus-Group-prev-unread-group`).

`N`
`C-n`
: Move point to the next newsgroup (`gnus-Group-next-group`).

`P`
`C-p`
: Move point to the previous newsgroup (`gnus-Group-prev-group`).

`j newsgroup RET`
: Move point to the specified *newsgroup* (`gnus-Group-jump-to-group`).

`/`
: Do an incremental search forward (`isearch-forward`).

`<`
: Move point to the beginning of the buffer (`beginning-of-buffer`).

`>`
: Move point to the end of the buffer (`end-of-buffer`).

`r`
: Restrict visible newsgroups to the current region specified by the point and the mark (`gnus-Group-restrict-groups`).

The command *`j`* (`gnus-Group-jump-to-group`) reads a newsgroup name interactively, and moves the point to it. If there is no such newsgroup in the buffer, a line for the newsgroup is inserted at the beginning of the buffer.

The command *`r`* (`gnus-Group-restrict-groups`) restricts visible newsgroups to the current region specified by the point and the mark. This is useful if you want to concentrate on a restricted set of newsgroups in a region. Type *`C-x w`* (`widen`) to remove the restriction.

4.2 Selecting a Newsgroup

If you successfully move the point to a desired newsgroup, you can select the newsgroup to read articles by typing *`SPC`* (`gnus-Group-read-group`) or `=` (`gnus-Group-select-group`) on that line.

`SPC`
: Select the newsgroup, and then select the first unread article automatically (`gnus-Group-read-group`).

`=` Select the newsgroup (`gnus-Group-select-group`). No article is selected automatically.

To prevent automatic selection of the first unread article even when the newsgroup is selected by the command *SPC* (`gnus-Group-read-group`), set the variable `gnus-auto-select-first` to `nil`. If you want to change the value of the variable according to the selected newsgroups, set the variable in the hook `gnus-Select-group-hook`.

If unread articles are contained in a selected newsgroup, they become ready to be read. Otherwise, all articles in the newsgroup become ready to be read. All articles in a newsgroup can be selected unconditionally by giving a prefix argument to the commands *SPC* and `=` (`gnus-Group-read-group` and `gnus-Group-select-group`).

If the number of articles being selected is larger than the variable `gnus-large-newsgroup`, the number of articles actually selected is asked for. If the given value *n* is positive, the last *n* articles will be selected. If *n* is negative, the first *n* articles will be selected. An empty string means to select all articles.

See Section A.1 [Variables], page 41, and see Section A.5 [Hooks], page 50, for more information on customization.

4.3 Maintaining Newsgroups

Subscription of newsgroups, deletion of bogus newsgroups, and other related operations are described in this section.

c Mark all articles as read in the newsgroup, preserving articles marked as unread (`gnus-Group-catch-up`).

C Mark all articles as read in the newsgroup (`gnus-Group-catch-up-all`).

l Redisplay newsgroups which are subscribed to and containing unread articles (`gnus-Group-list-groups`).

L Display all newsgroups unconditionally (`gnus-Group-list-all-groups`).

u Unsubscribe from (or subscribe to) the newsgroup (`gnus-Group-unsubscribe-current-group`).

U newsgroup RET
Unsubscribe from (or subscribe to) the specified *newsgroup* (`gnus-Group-unsubscribe-group`). If it is not contained in the startup file, it is added to the file.

C-k Kill newsgroups from the startup file (`gnus-Group-kill-group`). Numeric argument to the command specifies the number of newsgroups to be killed.

C-w Kill newsgroups in current region (excluding current line) from the startup file (`gnus-Group-kill-region`).

C-y Yank the last newsgroup killed with the command *C-k* before the current line (`gnus-Group-yank-group`).

C-c C-y Pop up a buffer for browsing the killed newsgroups (`gnus-Browse-killed-groups`).

b Check bogus newsgroups (`gnus-Group-check-bogus-groups`). Bogus means non-active.

g Get newly arrived articles (`gnus-Group-get-new-news`). In fact, GNUS reads the active file from the NNTP server again.

R Force to read the raw startup file and get newly arrived articles (`gnus-Group-restart`).

The commands *c* and *C* (`gnus-Group-catch-up` and `gnus-Group-catch-up-all`) mark all articles as read in a newsgroup. These commands do not take account of the cross-reference information in the '`Xref:`' field, while the command *c* (`gnus-Subject-catch-up-and-exit`) in Subject Mode does.

Only subscribed newsgroups containing unread articles are usually displayed in the Newsgroup buffer. Type *L* (`gnus-Group-list-all-groups`) to show all newsgroups which are currently active.

The command *U* (`gnus-Group-unsubscribe-group`) reads a newsgroup name interactively, and toggles its subscription flag if it is already in the startup file. Otherwise, if it is not contained in the startup file, it is added to the file. Thus, you can add newly created newsgroups manually which are not added automatically because of the options line in the startup file. See Section 1.4.6 [Startup File], page 14, for more information on the startup file and options line.

The commands *C-k* and *C-w* (`gnus-Group-kill-group` and `gnus-Group-kill-region`) kill newsgroups from both the Newsgroup buffer and the raw startup file. The killed newsgroups are stored in an invisible kill ring and they can be yanked back later. The command *C-y* (`gnus-Group-yank-group`) yank the last newsgroup killed. The command *C-c C-y* (`gnus-Browse-killed-groups`) pops up the *Browse-Killed buffer* for browsing all the killed newsgroups. In this buffer, the killed newsgroups can be yanked in any order using the commands *y* and *C-y* (`gnus-Browse-killed-yank`). Thus, you can change the order of newsgroups in the Newsgroup buffer without editing the raw startup file directly. Since the information on the killed newsgroups will be stored in the quick startup file, they can be restored any time unless you lose the file.

A *bogus newsgroup* is one not in the list of active newsgroups in the active file. Bogus newsgroups which are deleted or renamed must be deleted from the startup file (see Section 1.4.6 [Startup File], page 14) explicitly by the command *b* (`gnus-Group-check-bogus-groups`).

The command *R* (`gnus-Group-restart`) is useful to restart GNUS using the raw startup file instead of the quick startup file. Generally speaking, if you want changes to the raw startup file to be noticed by GNUS, it must be newer than the quick startup file. See Section 1.4.6 [Startup File], page 14, for more information on the startup file.

GNUS reads the active file at start up time to know about currently active articles. This information is not updated unless you force GNUS to do so with the command *g* (`gnus-Group-get-new-news`) or the command *R* (`gnus-Group-restart`).

4.4 Exiting GNUS

s
x Update the startup file `.newsrc` (`gnus-Group-force-update`).

z Suspend the current GNUS session to make it possible to resume it later (`gnus-Group-suspend`).

q Update the startup file `.newsrc`, and then exit GNUS (`gnus-Group-exit`).

Q Exit GNUS without updating the startup file `.newsrc` (`gnus-Group-quit`).

If a GNUS session is suspended by the command *z* (`gnus-Group-suspend`), it is possible to resume it later without any time-consuming initializations. Switch to the Newsgroup buffer and type *g* (`gnus-Group-get-new-news`) to get newly arrived articles if you want to resume the suspended GNUS session. It is a good idea to update the startup file (see Section 1.4.6 [Startup File], page 14) before suspending GNUS.

If you want to forget what you read this GNUS session, exit GNUS by the command *Q* (`gnus-Group-quit`). Otherwise, exit by the command *q* (`gnus-Group-exit`) to update the startup file.

The hook `gnus-Exit-gnus-hook` is called when exiting GNUS, and the hook `gnus-Suspend-gnus-hook` is called when suspending GNUS. If you want to clear out Emacs buffers which were created by GNUS and remain afterwards, use these hooks.

4.5 Miscellaneous Commands

Other miscellaneous commands are described here.

`a` Compose a new article (`gnus-Group-post-news`). See Section 5.6 [Followup and Reply], page 32, for more information.

M-k Edit a local KILL file (`gnus-Group-edit-local-kill`). See Chapter 7 [KILL File], page 37, for more information.

M-K Edit a global KILL file (`gnus-Group-edit-global-kill`). See Chapter 7 [KILL File], page 37, for more information.

V Print version number of this GNUS (`gnus-version`).

? Describe Group Mode commands briefly (`gnus-Group-describe-briefly`).

C-c C-i Read Info on Group Mode (`gnus-Info-find-node`). See Section 1.5 [Texinfo Manual], page 15, to prepare an Info file of GNUS.

5 Subject Commands

The Subject buffer is intended to show interesting headers of articles in a newsgroup and to help you know what kind of discussions are being held there. Messages of articles are displayed in the Article buffer which is usually popped up automatically when necessary. Scrolling of the messages and most other commands on articles can be done in the Subject buffer.

5.1 Reading Articles

5.1.1 Walking Around Headers

C-n Move point to the next header (`gnus-Subject-next-subject`).

C-p Move point to the previous header (`gnus-Subject-prev-subject`).

M-n Move point to the next header, skipping marked articles (`gnus-Subject-next-unread-subject`).

M-p Move point to the previous header, skipping marked articles (`gnus-Subject-prev-unread-subject`).

j number RET
Move point to the header specified by the article *number* (`gnus-Subject-goto-subject`).

/ Do an incremental search on headers (`isearch-forward`).

These commands are for moving the point on headers. Type *SPC* (`gnus-Subject-next-page`) or *g* (`gnus-Subject-show-article`) to read an article on the line (see Section 5.1.3 [Scrolling], page 26).

`=` Expand the Subject Mode window to show the headers full window (`gnus-Subject-expand-window`).

C-t Toggle truncation of the Subject buffer (`gnus-Subject-toggle-truncation`).

The command `=` (`gnus-Subject-expand-window`) expands the Subject Mode window by deleting the Article Mode window. It is very useful for displaying the Subject buffer full window when browsing through many headers. The command behaves different from the command *C-x 1* (`delete-other-windows`) if windows more than two are displayed.

Long lines in the Subject buffer are truncated and the continuation lines are not displayed normally. The command *C-t* (`gnus-Subject-toggle-truncation`) toggles the truncation of the lines in the buffer.

5.1.2 Moving Among Articles

The commands described here are for moving the point on headers, and then automatically selecting articles.

n Read the next article, skipping marked articles (`gnus-Subject-next-unread-article`).

p Read the previous article, skipping marked articles (`gnus-Subject-prev-unread-article`).

N Read the next article (`gnus-Subject-next-article`).

P Read the previous article (`gnus-Subject-prev-article`).

M-C-n Read the next article with the same subject as the current article (`gnus-Subject-next-same-subject`).

M-C-p Read the previous article with the same subject as the current article (`gnus-Subject-prev-same-subject`).

M-x gnus-Subject-next-unread-same-subject
Read the next article with the same subject as the current article, skipping marked articles.

M-x gnus-Subject-prev-unread-same-subject
Read the previous article with the same subject as the current article, skipping marked articles.

. Read the first unread article (`gnus-Subject-first-unread-article`).

l Read the article selected last (`gnus-Subject-goto-last-article`).

J number RET
Read the article specified by the article *number* (`gnus-Subject-goto-article`).

If the variable `gnus-auto-select-same` is non-`nil`, the commands *n* and *p* (`gnus-Subject-next-unread-article` and `gnus-Subject-prev-unread-article`) move the point to unread articles with the same subject as the current article like the commands *M-x gnus-Subject-next-unread-same-subject* and *M-x gnus-Subject-prev-unread-same-subject*, respectively. If you are used to running '`rn -S`', set the variable to non-`nil`.

If the variable `gnus-auto-extend-newsgroup` is non-`nil`, the commands *N* and *P* (`gnus-Subject-next-article` and `gnus-Subject-prev-article`) extend visible articles to forward and backward if possible.

The variable `gnus-auto-select-next` defines the behavior of GNUS when there is no unread article in the current newsgroup and a command selecting the next unread article is executed. If the variable is non-`nil`, the next newsgroup containing unread articles is selected automatically.

See Section A.1 [Variables], page 41, for more information on customization.

5.1.3 Scrolling Within an Article

Type *SPC* (`gnus-Subject-next-page`) to scroll to the next page of the current article. If no article is selected yet, an article near the point is selected and its first page is displayed in the Article buffer. The next unread article is selected automatically if *SPC* is typed at the end of the message.

SPC Scroll to the next page of the current article (`gnus-Subject-next-page`). Select it first if no article is selected yet. Select the next unread article automatically at the end of the message.

`DEL`	Scroll to the previous page of the current article (`gnus-Subject-prev-page`).
`RET`	Scroll up or down one line of the current article (`gnus-Subject-scroll-up`).
`<`	Move point to the beginning of the current article (`gnus-Subject-beginning-of-article`).
`>`	Move point to the end of the current article (`gnus-Subject-end-of-article`).
`w`	Stop page breaking (`gnus-Subject-stop-page-breaking`).
`v`	Show all headers of the current article (`gnus-Subject-show-all-headers`).
`t`	Show all headers of the current article if pruned header currently shown, or vice versa (`gnus-Subject-toggle-header`).
`C-c C-r`	Caesar rotate letters by 13 places and Japanese characters by 47 places (`gnus-Subject-caesar-message`).
`g`	Force to read the current article again (`gnus-Subject-show-article`).

If the Article buffer is not visible, it is popped up under the Subject buffer when necessary. The height of the Subject buffer and that of the Article buffer can be customized by using the variable `gnus-window-configuration`.

The command *`C-c C-r`* (`gnus-Subject-caesar-message`) rotates all letters in the message body of the current article by 13 places. Japanese characters are rotated by 47 places. Running the command twice on the same article results the original message.

If the variable `gnus-break-pages` is non-`nil`, the message is broken into pages at page delimiters specified by the variable `gnus-page-delimiter`. The command *`w`* (`gnus-Subject-stop-page-breaking`) temporary suspends page breaks.

The variable `gnus-ignored-headers` specifies header fields which should be ignored. The command *`v`* (`gnus-Subject-show-all-headers`) shows all headers of the current article, while the command *`t`* (`gnus-Subject-toggle-header`) toggles the headers.

See Section A.1 [Variables], page 41, and see Section A.5 [Hooks], page 50, for more information on customization.

5.1.4 Marking Articles

GNUS uses three kinds of marks to indicate article status.

- White space ‘ ’ for newly arrived articles.
- Dash ‘`-`’ for articles marked as unread.
- Any other characters for articles marked as read.

The status is displayed at the beginning of each line of the Subject buffer. Commands for marking or removing these marks are as follows:

`d`	Mark article as read, and then move to the next subject (`gnus-Subject-mark-as-read-forward`).
`D`	Mark article as read, and then move to the previous subject (`gnus-Subject-mark-as-read-backward`).
`u`	Mark article as unread, and then move to the next subject (`gnus-Subject-mark-as-unread-forward`).

U Mark article as unread, and then move to the previous subject (`gnus-Subject-mark-as-unread-backward`).

M-u Clear marks, and then move to the next subject (`gnus-Subject-clear-mark-forward`).

M-U Clear marks, and then move to the previous subject (`gnus-Subject-clear-mark-backward`).

k Mark articles with the same subject as the current article as read, and then select the next unread article (`gnus-Subject-kill-same-subject-and-select`).

C-k Mark articles with the same subject as the current article as read (`gnus-Subject-kill-same-subject`).

c
M-x gnus-Subject-catch-up-and-exit
Mark all articles, which are not marked as unread, as read, and then exit the current newsgroup.

M-x gnus-Subject-catch-up-all-and-exit
Mark all articles as read, and then exit the current newsgroup.

M-x gnus-Subject-catch-up
Mark all articles as read, preserving articles marked as unread.

M-x gnus-Subject-catch-up-all
Mark all articles as read.

It is helpful to delete headers marked as read while reading a large newsgroup. The command *x* (`gnus-Subject-delete-marked-as-read`) deletes headers marked as read. The command *X* (`gnus-Subject-delete-marked-with`) deletes headers which have a specific mark.

x Delete headers marked as read (`gnus-Subject-delete-marked-as-read`).

X mark RET
Delete headers marked with *mark* (`gnus-Subject-delete-marked-with`).

5.1.5 Reading Based on Conversation Threads

Conversations on the usenet news usually contain several threads under a single subject. This makes it difficult to know which article follows which without reading references directly. It would be useful if we could read articles based on conversation threads.

GNUS enables you this thread-based reading. The reader can follow threads of conversation, mark threads as read, and go up and down thread trees. The command *M-C-t* (`gnus-Subject-toggle-threads`) toggles showing conversation threads in Subject Mode. If it is turned on, Subject buffer is displayed in a tree structured form according to what each article was a reply to.

M-C-t Toggle thread-based reading (`gnus-Subject-toggle-threads`).

M-C-s Show thread subtrees (`gnus-Subject-show-thread`).

M-x gnus-Subject-show-all-threads
Show all thread subtrees.

M-C-h Hide thread subtrees (`gnus-Subject-hide-thread`).

M-x gnus-Subject-hide-all-threads
Hide all thread subtrees.

M-C-f Go to the same level next thread (`gnus-Subject-next-thread`).

M-C-b Go to the same level previous thread (`gnus-Subject-prev-thread`).

M-C-d Go down to the lower level thread (`gnus-Subject-down-thread`).

M-C-u Go up to the upper level thread (`gnus-Subject-up-thread`).

M-C-k Mark articles under current thread as read (`gnus-Subject-kill-thread`).

Thread subtrees can be hidden by using the command *M-C-h* (`gnus-Subject-hide-thread`), and the hidden subtrees can be shown by using the command *M-C-s* (`gnus-Subject-show-thread`).

If the variable `gnus-thread-hide-killed` is non-`nil`, thread subtrees killed by the command *M-C-k* (`gnus-Subject-kill-thread`) are hidden automatically.

If you want to hide thread subtrees initially, set the variable `gnus-thread-hide-subtree` to non-`nil`.

If you want to enable thread-based reading automatically, set the variable `gnus-show-threads` to non-`nil`.

See Section A.1 [Variables], page 41, for more information on customization.

5.1.6 Reading Digest Articles

Digest article is a message containing many messages in *digest* format. Since a digest article contains many messages in one article, it is a little bit difficult to read it on a per message basis. The following commands make it easier to read each message in a digest.

C-c C-n Scroll to the next digest message of the current article (`gnus-Subject-next-digest`).

C-c C-p Scroll to the previous digest message of the current article (`gnus-Subject-prev-digest`).

C-d Read the current digest article using Rmail (`gnus-Subject-rmail-digest`).

The commands *C-c C-n* and *C-c C-p* (`gnus-Subject-next-digest` and `gnus-Subject-prev-digest`) scroll a digest article to the next and previous digested message, respectively. The variable `gnus-digest-separator` specifies a regexp which separates digested messages.

The command *C-d* (`gnus-Subject-rmail-digest`) runs Rmail on a digest article and makes it possible to read messages not in digest form using Rmail Mode. See Info file `emacs`, node '`Rmail`', for more information on Rmail Mode. Use the hook `gnus-Select-article-hook` to run Rmail on digest articles automatically.

Digest articles in some newsgroups of USENET cannot be read using Rmail. In this case, a message '`Article is not a digest`' is displayed in the echo area of Emacs. This means that these articles are not written in the proper digest format. It is, however, possible to read these incomplete digest articles by modifying the message headers or bodies appropriately using the hook `gnus-Select-digest-hook`. See Section A.5 [Hooks], page 50, to modify incomplete digest articles.

If the variable `gnus-digest-show-summary` is non-`nil`, a summary of the digest article is also displayed automatically when Rmail is invoked.

5.2 Searching Articles

`s`
: Do incremental search on the current article (`gnus-Subject-isearch-article`).

`M-s regexp RET`
: Search for articles containing a match for *regexp* forward (`gnus-Subject-search-article-forward`). If *regexp* is empty, the last regexp used is used again.

`M-S regexp RET`
: Search for articles containing a match for *regexp* backward (`gnus-Subject-search-article-backward`). If *regexp* is empty, the last regexp used is used again.

`& field RET regexp RET command RET`
: Execute *command* on articles containing a match for *regexp* in *field* of the headers (`gnus-Subject-execute-command`). If *field* is empty, the entire article is searched for.

The command *`s`* (`gnus-Subject-isearch-article`) does an incremental search on the current article. The commands *`M-s`* and *`M-S`* (`gnus-Subject-search-article-forward` and `gnus-Subject-search-article-backward`) search for articles containing a match for regexp. The search starts from the current point of the current article.

The command *`&`* (`gnus-Subject-execute-command`) interactively reads the field name of article headers, regexp, and a valid command key sequence. It then searches for articles having a field that contains a match for the regexp, and then executes the command specified by the key sequence on them.

5.3 Referencing Articles

`^`
: Refer to parent of the current article (`gnus-Subject-refer-parent-article`). With a prefix argument, go back to the child.

`M-r Message-ID RET`
: Refer to the article by using the *Message-ID* (`gnus-Subject-refer-article`). With an empty *Message-ID*, go back to the origin.

The command *`^`* (`gnus-Subject-refer-parent-article`) refers to parent article of the current article. The child article is remembered in internal history, and it is possible to return to the child by giving a prefix argument to the command.

The commands *`^`* and *`M-r`* (`gnus-Subject-refer-parent-article` and `gnus-Subject-refer-article`) share the same internal history. You can thus go back to a child using the command *`M-r`* with an empty Message-ID.

Type *`g`* (`gnus-Subject-show-article`) to go back to the origin from the visited articles directly.

See Chapter 6 [Article Commands], page 35, to refer to articles by using Message-IDs included in the messages.

5.4 Saving Articles

GNUS provides four different formats for saving articles: Rmail format, Unix mailbox format, MH folder, and article format. If you set the variable `gnus-default-article-saver` to your favorite article saver, you can save an article using the command *`o`* (`gnus-Subject-save-article`) in your favorite format. The default saver is the function `gnus-Subject-save-in-rmail`.

`o`
: Save the current article using the default saver specified by the variable `gnus-default-article-saver` (`gnus-Subject-save-article`).

`C-o`
`M-x gnus-Subject-save-in-mail`
: Save the current article in Unix mailbox format.

`M-x gnus-Subject-save-in-rmail`
: Save the current article in Rmail format.

`M-x gnus-Subject-save-in-folder`
: Save the current article in an MH folder.

`M-x gnus-Subject-save-in-file`
: Save the current article in article format.

`| command RET`
: Send contents of the current article to the *command* subprocess (`gnus-Subject-pipe-output`).

If the variable `gnus-save-all-headers` is non-`nil`, all headers of an article are saved.

The variable `gnus-article-save-directory` specifies a directory in which articles are saved to by the functions `gnus-Subject-save-in-mail`, `gnus-Subject-save-in-rmail`, and `gnus-Subject-save-in-file`. It is initialized from the `SAVEDIR` environment variable. The default directory is ‘`~/News`’.

The variables `gnus-mail-save-name`, `gnus-rmail-save-name`, `gnus-folder-save-name`, and `gnus-file-save-name` specify functions generating default file name to which articles are saved using `gnus-Subject-save-in-mail`, `gnus-Subject-save-in-rmail`, `gnus-Subject-save-in-folder`, and `gnus-Subject-save-in-file`, respectively. The function is called with *newsgroup*, *headers*, and optional *last-name*.

See Section A.1 [Variables], page 41, for more information on customization.

5.5 Sorting Headers

The headers listed in the Subject buffer can be sorted by number, subject, date, or author of articles. Sorting is stable, so it is possible to combine them to sort the headers with multiple keys. To sort in reverse order, give a prefix argument to the commands.

`C-c C-s n`
`C-c C-s C-n`
: Sort the headers by number (`gnus-Subject-sort-by-number`).

`C-c C-s s`
`C-c C-s C-s`
: Sort the headers by subject (`gnus-Subject-sort-by-subject`).

C-c C-s d
C-c C-s C-d
  Sort the headers by date (`gnus-Subject-sort-by-date`).

C-c C-s `a`
C-c C-s C-`a`
  Sort the headers by author (`gnus-Subject-sort-by-author`).

It is also possible to sort the headers automatically when a newsgroup is selected using the hook `gnus-Select-group-hook` (see Section A.5 [Hooks], page 50).

5.6 Followup and Reply

f Followup to the current article (`gnus-Subject-post-reply`).

F Followup to the current article with the original article (`gnus-Subject-post-reply-with-original`).

`a` Compose a new article (`gnus-Subject-post-news`).

Type `a` (`gnus-Subject-post-news`) to post a new article. If the variable `gnus-interactive-post` is non-`nil`, the newsgroup, subject, and distribution are asked for interactively. The command *f* (`gnus-Subject-post-reply`) fills these values in automatically from those of the original article. Type *C-c C-y* (`news-reply-yank-original`) to include the original article. The command *F* (`gnus-Subject-post-reply-with-original`) yanks the original article automatically. If you want to followup to several articles in a single article and want to include them in it, type *F* for each of them. You will be asked if a text being edited should be erased. You should answer '`n`' to the question.

If the variable `gnus-novice-user` is non-`nil`, your confirmations will be required for composing a new article.

The major mode for composing a new article is *News Mode* which is borrowed from `rnewspost.el`. Type *C-h m* (`describe-mode`) to get more help on News Mode.

C Cancel the current article you posted (`gnus-Subject-cancel-article`).

Suppose you post an article and then later realize that you made a horrible mistake. You really do not want anyone to see your article. You want the article to be removed from any machines that it may have reached. The command *C* (`gnus-Subject-cancel-article`) is intended to do this. First select the offending article as current, then type *C*.

r Reply to the author of the current article (`gnus-Subject-mail-reply`).

R Reply to the author of the current article with the original article (`gnus-Subject-mail-reply-with-original`).

C-c C-f Forward the current message to another user. (`gnus-Subject-mail-forward`).

m Compose a mail message in other window (`gnus-Subject-mail-other-window`).

Use the command *r* (`gnus-Subject-mail-reply`) to reply to the author of the article. Type *C-c C-y* to include the original article. The command *R* (`gnus-Subject-mail-reply-with-original`) yanks the original article automatically.

When composing a mail message, the message composer (or mailer) is selected by the values of the variables `gnus-mail-reply-method`, `gnus-mail-forward-method`, and `gnus-mail-other-window-method`. These are defaulted to use Mail Mode. If you want to use mh-e letter Mode instead of it, set the variable `gnus-mail-reply-method` to `gnus-mail-reply-using-mhe`, variable `gnus-mail-forward-method` to `gnus-mail-forward-using-mhe`, and the variable `gnus-mail-other-window-method` to `gnus-mail-other-window-using-mhe`. It is possible to use other mailers by customizing these variables. See Info file `emacs`, node '`Mail Mode`', for more information on Mail Mode.

5.7 Exiting the Current Newsgroup

q Exit the current newsgroup, and return to Group Mode (`gnus-Subject-exit`).

Q Exit the current newsgroup without recording unread articles information (`gnus-Subject-quit`).

c
M-x gnus-Subject-catch-up-and-exit
Mark all articles, which are not marked as unread, as read, then exit the current newsgroup.

M-x gnus-Subject-catch-up-all-and-exit
Mark all articles as read, then exit the current newsgroup.

G Record unread articles information, then reselect the current newsgroup (`gnus-Subject-reselect-current-group`).

M-x gnus-Subject-next-group
Record unread articles information, then select the next newsgroup containing unread articles.

M-x gnus-Subject-prev-group
Record unread articles information, then select the previous newsgroup containing unread articles.

The command *G* (`gnus-Subject-reselect-current-group`) selects the current newsgroup again after temporary exiting the newsgroup. If no articles remain unread, all articles in the newsgroup will be selected. A prefix argument to the command means to select all articles in the newsgroup.

5.8 Miscellaneous Commands

Other miscellaneous commands are described here.

M-k Edit a local KILL file applied to the current newsgroup (`gnus-Subject-edit-local-kill`). See Chapter 7 [KILL File], page 37, for more information.

M-K Edit a global KILL file applied to all newsgroups (`gnus-Subject-edit-local-kill`). See Chapter 7 [KILL File], page 37, for more information.

V Print the version number of this GNUS (`gnus-version`).

? Describe Subject Mode commands briefly (`gnus-Subject-describe-briefly`).

C-c C-i Read Info on Subject Mode (`gnus-Info-find-node`). See Section 1.5 [Texinfo Manual], page 15, to prepare an Info file of GNUS.

6 Article Commands

In the Article buffer the following commands are available:

SPC Scroll text of the current window (`gnus-Article-next-page`).

DEL Scroll text of the current window (`gnus-Article-prev-page`).

r Refer to article specified by the Message-ID close to the point (`gnus-Article-refer-article`).

o Return to the previous article from the referenced article (`gnus-Article-pop-article`).

h
s Reconfigure Emacs windows to show the Subject buffer above the Article buffer and move the point to the Subject buffer (`gnus-Article-show-subjects`).

? Describe Article Mode commands briefly (`gnus-Article-describe-briefly`).

C-c C-i Read Info on Article Mode (`gnus-Info-find-node`). See Section 1.5 [Texinfo Manual], page 15, to prepare an Info file of GNUS.

The command *r* (`gnus-Article-refer-article`) searches for the Message-ID around the point, and refers to the article specified by it if found. Use the command *o* (`gnus-Article-pop-article`) to return to the previous article. See Section 5.3 [Referencing Articles], page 30, for referencing parent articles easily.

7 KILL File

7.1 What KILL Files Do

A *KILL* file contains lisp expressions to be applied to a selected newsgroup. The purpose is to mark articles as read on the basis of some set of regexps.

There are two kinds of KILL files, global and local. A global KILL file is applied to every newsgroup, and a local KILL file to a specified newsgroup. Since a global KILL file is applied to every newsgroup, for better performance use a local one.

7.2 Making a KILL File

A KILL file can contain any kind of Emacs lisp expressions expected to be evaluated in the Subject buffer. Writing lisp programs for this purpose is not easy because the internal working of GNUS must be well-known. For this reason, GNUS provides a general function which does this easily for non-lisp programmers.

```
(gnus-kill field regexp &optional command all)
```

The `gnus-kill` function executes commands available in Subject Mode by their key sequences. `gnus-kill` must be called with *field*, *regexp*, and optional *command* and *all*. *Field* is a string representing the header field or an empty string. If *field* is an empty string, the entire article body is searched for. *Regexp* is a string which is compared with *field* value. *Command* is a string representing a valid key sequence in Subject Mode or a lisp expression. *Command* is default to `(gnus-Subject-mark-as-read nil "X")`. Make sure that *command* is executed in the Subject buffer. If the second optional argument *all* is non-`nil`, the *command* is applied to articles which are already marked as read or unread. Articles which are marked are skipped over by default.

For example, if you want to mark articles of which subjects contain the string '`AI`' as read, a possible KILL file may look like:

```
(gnus-kill "Subject" "AI")
```

If you want to mark articles with '`D`' instead of '`X`', you can use the following expression:

```
(gnus-kill "Subject" "AI" "d")
```

In this example it is assumed that the command `gnus-Subject-mark-as-read-forward` is assigned to *`d`* in Subject Mode.

It is possible to delete unnecessary headers which are marked with '`X`' in a KILL file by using the function `gnus-expunge` as follows:

```
(gnus-expunge "X")
```

If the Subject buffer is empty after applying KILL files, GNUS will exit the selected newsgroup normally. If headers which are marked with '`D`' are deleted in a KILL file, it is impossible to read articles which are marked as read in the previous GNUS sessions. Marks other than '`D`' should be used for articles which should really be deleted.

All sorts of searches in Subject Mode normally ignore the case of the text they are searching through. If you do not want to ignore the case, set the variable `case-fold-search` to `nil`.

7.3 Editing KILL Files

The command *M-K* in Subject Mode and Group Mode (`gnus-Subject-edit-global-kill` and `gnus-Group-edit-global-kill`) pops up an Emacs buffer for editing a global KILL file. A global KILL file is created in the directory specified by the variable `gnus-article-save-directory` (default to `~/News`), and its file name is specified by the variable `gnus-kill-file-name` (default to `KILL`).

The command *M-k* in Subject Mode and Group Mode (`gnus-Subject-edit-local-kill` and `gnus-Group-edit-local-kill`) pops up an Emacs buffer for editing a local KILL file. A local KILL file for a newsgroup *news.group* is created as *news.group*`.KILL` in the directory specified by the variable `gnus-article-save-directory` if the variable `gnus-use-long-file-name` is non-`nil`. Otherwise, if the variable `gnus-use-long-file-name` is `nil`, the file is created as *news/group/*`KILL` under the same directory.

The major mode of these buffers is *KILL-File Mode*. This mode is specialized for editing Emacs lisp programs the same as Emacs-Lisp Mode. In addition to Emacs-Lisp Mode, the following commands are available:

C-c C-k C-s
: Insert a template of a kill command on subject (`gnus-Kill-file-kill-by-subject`).

C-c C-k C-a
: Insert a template of a kill command on author (`gnus-Kill-file-kill-by-author`).

C-c C-a
: Apply current buffer being edited to selected newsgroup (`gnus-Kill-file-apply-buffer`).

C-c C-e
: Apply sexp before point in current buffer to selected newsgroup (`gnus-Kill-file-apply-last-sexp`).

C-c C-c
: Save the KILL file and then return to the previous buffer (`gnus-Kill-file-exit`).

C-c C-i
: Read Info on KILL file (`gnus-Info-find-node`). See Section 1.5 [Texinfo Manual], page 15, to prepare an Info file of GNUS.

If KILL-File Mode is invoked from Subject Mode by the command `gnus-Subject-edit-local-kill` or `gnus-Subject-edit-global-kill`, the commands *C-c C-k C-s* and *C-c C-k C-a* (`gnus-Kill-file-kill-by-subject` and `gnus-Kill-file-kill-by-author`) insert a kill command on the subject and author of an article where the point is on, respectively. Otherwise, a template of a kill command is inserted.

The commands *C-c C-a* and *C-c C-e* (`gnus-Kill-file-apply-buffer` and `gnus-Kill-file-apply-last-sexp`) can be used to test kill commands being edited in current buffer. The kill commands are applied to current newsgroup.

7.4 Example of a KILL File

The following is an example of a local KILL file for newsgroup ‘`control`’. This is currently being used by the author.

```
;; Apply to the newsgroup `control' if the NNTP server is flab.
```

```
(if (string-equal gnus-nntp-server "flab")
    (progn
      (gnus-kill "Subject" "ihave flab\\|sendme")
      (gnus-kill "Subject" "cancel\\|newgroup\\|rmgroup" "d")
      (gnus-expunge "X")))
```

7.5 Background Kill Processing

Kill processing may take long time. If it becomes terribly frustrating, try background kill processing using the following shell command:

```
emacs -batch -l gnus -f gnus-batch-kill NEWSGROUPS
```

where *NEWSGROUPS* argument is newsgroup names separated by either white spaces or a comma. '`!`' preceding a newsgroup name means negation, and '`all`' matches anything else. These interpretations are the same as the options line of the startup file (see Section 1.4.6 [Startup File], page 14).

7.6 Advanced Kill Processing

Internally, applying kills means to run the hook `gnus-Apply-kill-hook`. It is called after the Subject buffer is prepared for a selected newsgroup. The default hook is the function `gnus-apply-kill-file` which loads a global KILL file and a local KILL file in this order. A different style of the kill processing can be implemented by customizing this hook.

For example, if you think a global KILL file is unnecessary, you can use the following hook which applies only a local KILL file. This change can save the time for checking the existence of a global KILL file.

```
(setq gnus-Apply-kill-hook
      (function
       (lambda ()
         ;; Apply a local KILL file.
         (load (gnus-newsgroup-kill-file gnus-newsgroup-name) t nil t))))
```

On the contrary, the following example enables only a global KILL file.

```
(setq gnus-Apply-kill-hook
      (function
       (lambda ()
         ;; Apply a global KILL file.
         (load (gnus-newsgroup-kill-file nil) t nil t))))
```

Here is an advanced example that drastically reduces the time for applying KILL files. This hook does the kill processing directly without loading the KILL files.

```
(setq gnus-Apply-kill-hook
      (function
       (lambda ()
         ;; Apply to the newsgroup `control'
         ;; if the NNTP server is flab.
         (and (string-equal gnus-nntp-server "flab")
              (string-equal gnus-newsgroup-name "control")
              (progn
                (gnus-kill "Subject" "ihave flab\\|sendme")
                (gnus-kill "Subject" "cancel\\|newgroup\\|rmgroup" "d")
                (gnus-expunge "X"))))))
```

Appendix A Customizing GNUS

Appendix A describes the variables and hooks for simple customization and the variables for localization.

A.1 Common Variables

`gnus-nntp-server`
: Specifies the name of the host running the NNTP server. The variable is initialized from the `NNTPSERVER` environment variable. If the server name is preceded by a colon such as '`:Mail`', the user's private directory `~/Mail` is used as a news spool. See Section 1.4.1 [NNTP Server], page 12, and see Section 2.3 [Private Directory], page 17, for more information.

`gnus-nntp-service`
: Specifies a service name of NNTP, usually a string `"nntp"`. In a few instances, it must be the number `119`. To use a local news spool of your machine rather than NNTP, set the variable to `nil`. See Section 1.4.2 [NNTP Service], page 13, and see Section 2.2 [Local News Spool], page 17, for more information.

`gnus-local-domain`
: Specifies the domain which is the domain part of your mail address excluding the local host name of your machine. The environment variable `DOMAINNAME` is used instead if defined. If the function `system-name` returns the full Internet name, there is no need to define the domain. See Section 1.4.3 [Domain and Organization], page 13, for more information.

`gnus-local-organization`
: Specifies the organization you belong to. The environment variable `ORGANIZATION` is used instead if defined. If the value begins with a slash, it is taken as the name of a file whose contents are read for the value. See Section 1.4.3 [Domain and Organization], page 13, for more information.

`gnus-local-timezone`
: Specifies the local time zone you belong to. The value can be either a time zone name such as '`"JST"`' or a difference in hour from GMT such as '`+0900`'. If the variable is non-`nil`, a general time zone handling package `timezone.el` is used to generate a valid date for '`Date:`' field in terms of RFC822. Otherwise, if it is nil, GNUS generate a date ignoring the local time zone. If you are using Bnews, it is okay since `inews` will rewrite the invalid date. However, if you are using Cnews, you must set the variable to the correct time zone or set the variable `gnus-news-system` to '`Cnews`' since `inews` of Cnews does not rewrite the wrong '`Date:`' field.

`gnus-use-generic-from`
: Non-`nil` means the local host name of your machine will not appear in the '`From:`' field of article headers. If the variable is a string, it is used as your domain instead of the definition by the variable `gnus-local-domain` or the environment variable `DOMAINNAME`. See Section 1.4.4 [GENERICFROM], page 13, for more information.

`gnus-use-generic-path`
: Non-`nil` means the NNTP server name will not appear in the '`Path:`' field of article headers. If the variable is a string, it is used in the '`Path:`' field as the NNTP server name instead of the definition by the variable `gnus-nntp-server`. See Section 1.4.5 [GENERICPATH], page 14, for more information.

`gnus-news-system`
: Specifies news software system name of the news server, such as Bnews and Cnews. It is intended to hide implementation dependent differences between news systems. Current version understands '`Bnews`' and '`Cnews`'.

`gnus-startup-file`
: Specifies a startup file of the Bnews system, usually `.newsrc`. If there is a file named `.newsrc-`*`server`*, it is used instead when talking to *server*. See Section 1.4.6 [Startup File], page 14, for more information.

`gnus-signature-file`
: Specifies a signature file of the Bnews system, usually `.signature`. If there is a file named `.signature-`*`distribution`*, it is used instead when posting an article in *distribution*. Set the variable to `nil` to prevent appending the signature file automatically.

`gnus-use-cross-reference`
: Specifies what to do with cross references ('`Xref:`' field). If it is `nil`, cross references are ignored. If it is `t`, articles in subscribed newsgroups are only marked as read. Otherwise, if it is not `nil` nor `t`, articles in all newsgroups are marked as read.

`gnus-use-followup-to`
: Specifies what to do with '`Followup-To:`' field. If it is `nil`, its value is ignored. If it is non-`nil`, its value is used as followup newsgroups. Especially, if it is `t` and you are going to followup to an article in which `poster` is specified, your confirmation is required.

`gnus-use-full-window`
: Non-`nil` means to take up the entire screen of Emacs. If the variable is `nil`, the windows used by GNUS will be restricted to the bounds of the original window. This is very useful if you want to read articles while you do other work in other windows.

`gnus-window-configuration`
: Specifies the configuration of the Group Mode window, the Subject Mode window, and the Article Mode window. The window configuration can be specified for each action of GNUS (e.g. selecting a newsgroup or selecting an article). This is quite useful if you are using a slow terminal since the update of Emacs windows can be minimized by displaying these three windows same time.

  The variable must be a list of '`(`*`action`*` (`*`g s a`*`))`', where *action* is an action being performed, and *g*, *s*, and *a* are the relative heights of the Group Mode window, the Subject Mode window, and the Article Mode window, respectively. *Action* is '`SelectNewsgroup`', '`ExitNewsgroup`', '`SelectArticle`', or '`ExpandSubject`'.

The following example is the default window configuration:

```
(setq gnus-window-configuration
      '((SelectNewsgroup (0 1 0))
        (ExitNewsgroup   (1 0 0))
        (SelectArticle   (0 3 10))
        (ExpandSubject   (0 1 0))))
```

The following is an example of yet another two windows mode. Article buffer is always displayed on a screen. This is useful on a slow terminal.

```
(setq gnus-window-configuration
      '((SelectNewsgroup (0 1 0))
        (ExitNewsgroup   (1 0 3))
        (SelectArticle   (0 1 3))
        (ExpandSubject   (0 1 0))))
```

The following is an example of three windows mode. Three buffers are always displayed on a screen. This is also useful on a slow terminal.

```
(setq gnus-window-configuration
      '((SelectNewsgroup (1 4 0))
        (ExitNewsgroup   (1 1 3))
        (SelectArticle   (1 1 3))
        (ExpandSubject   (1 4 0))))
```

`gnus-large-newsgroup`
: Specifies the number of the articles which indicates a large newsgroup. If the number of articles in a newsgroup is greater than this value, the number of articles to be selected is asked for. If the given value *n* is positive, the last *n* articles are selected. If *n* is negative, the first *n* articles are selected. An empty string means to select all articles.

`gnus-author-copy`
: Specifies a file name saving a copy of an article posted using '`FCC:`' field. The variable is initialized from the `AUTHORCOPY` environment variable. The specified file name is inserted in '`FCC:`' field, so you have a chance to change the file name or disable saving a copy by editing this field.

If the first character of the value is not `` `|' ``, the article is saved to the specified file using the function specified by the variable `gnus-author-copy-saver`. The default function `rmail-output` saves in Unix mailbox format. Instead, if the first character is `` `|' ``, the contents of the article is send to a program specified by the rest of the value. For example, articles can be saved in an MH folder by the following:

```
(setq gnus-author-copy
      "|/usr/local/lib/mh/rcvstore +Article")
```

`gnus-author-copy-saver`
: Specifies a function to save an author copy to. The function is called with a file name to save a copy to. The default function `rmail-output` saves in Unix mailbox format.

`gnus-use-long-file-name`
: Non-`nil` means that a newsgroup name is used as a default file name to save articles to. If it is `nil`, the directory form of a newsgroup name is used instead. It is set to nil by default if the variable `system-type` is either '`usg-unix-v`' or '`xenix`'.

`gnus-mail-save-name`
`gnus-rmail-save-name`
`gnus-folder-save-name`
`gnus-file-save-name`
: Specifies a function generating a file name to save articles to. The function is called with *newsgroup*, *headers*, and optional *last-name*. *Newsgroup* is a string representing the current newsgroup name. *Headers* is a vector containing headers of the current article. Macros and functions accessing contents of the *headers* are defined as `nntp-header-`*`field`* and `gnus-header-`*`field`*, respectively. The following functions are provided as file name generators by default:

  `gnus-numeric-save-name`
  : Return a file name like '*`news.group/number`*' or '*`news/group/number`*' according to the variable `gnus-use-long-file-name`.

  `gnus-Numeric-save-name`
  : Return a file name like '*`News.group/number`*' or '*`news/group/number`*' according to the variable `gnus-use-long-file-name`.

  `gnus-plain-save-name`
  : Return a file name like '*`news.group`*' or '*`news/group/`*`news`' according to the variable `gnus-use-long-file-name`.

  `gnus-Plain-save-name`
  : Return a file name like '*`News.group`*' or '*`news/group/`*`news`' according to the variable `gnus-use-long-file-name`.

  `gnus-folder-save-name`
  : Return a folder name like '`+`*`news.group`*' or '`+`*`news/group`*' according to the variable `gnus-use-long-file-name`.

  `gnus-Folder-save-name`
  : Return a folder name like '`+`*`News.group`*' or '`+`*`news/group`*' according to the variable `gnus-use-long-file-name`.

`gnus-default-article-saver`
: Specifies a function to save articles in your favorite format using the command `gnus-Subject-save-article`. The function must be interactively funcallable. In other words, it must be an Emacs command. The functions currently provided are as follows:

  `gnus-Subject-save-in-mail`
  : Save articles in Unix mailbox format.

`gnus-Subject-save-in-rmail`
: Save articles in Rmail format.

`gnus-Subject-save-in-folder`
: Save articles in an MH folder.

`gnus-Subject-save-in-file`
: Save articles in article format.

`gnus-article-save-directory`
: Specifies a directory name to save articles in using the commands `gnus-Subject-save-in-mail`, `gnus-Subject-save-in-rmail`, and `gnus-Subject-save-in-file`. The variable is initialized from the `SAVEDIR` environment variable. Its default value is `~/News`.

`gnus-kill-file-name`
: Specifies a file name of KILL file (see Chapter 7 [KILL File], page 37). Its default value is `KILL`.

`gnus-default-distribution`
: Specifies a distribution inserted automatically when no distribution is specified.

`gnus-novice-user`
: Non-`nil` means you are a novice to USENET. If it is non-`nil`, verbose messages may be displayed or your confirmations may be required.

`gnus-interactive-post`
: Non-`nil` means that newsgroup, subject, and distribution are asked for interactively when composing a new article.

`gnus-user-login-name`
: Specifies your login name. The login name is got from the `USER` and `LOGNAME` environment variables and the function `user-login-name`, if undefined.

`gnus-user-full-name`
: Specifies your full name. The full name is got from the `NAME` environment variable and the function `user-full-name`, if undefined.

`gnus-show-all-headers`
: Non-`nil` means all headers of an article are shown.

`gnus-save-all-headers`
: Non-`nil` means all headers of an article are saved in a file.

`gnus-show-threads`
: Non-`nil` means conversation threads are displayed in a tree structured form according to references in Subject Mode.

`gnus-thread-hide-subject`
: Non-`nil` means subjects of lower level threads are hidden if the thread-based reading is turned on.

`gnus-thread-hide-subtree`
: Non-`nil` means thread subtrees are hidden initially. If thread subtrees are hidden, you have to run the command `gnus-Subject-show-thread` by hand or by using `gnus-Select-article-hook` to show them.

`gnus-thread-hide-killed`
: Non-`nil` means killed thread subtrees are hidden automatically.

`gnus-thread-ignore-subject`
: Non-`nil` means subject differences are ignored but only references are taken into account in constructing threads trees. If it is non-`nil` and thread subtrees are hidden, some commands work with subjects may not work properly.

`gnus-thread-indent-level`
: Specifies indentation level of thread subtrees.

`gnus-auto-extend-newsgroup`
: Non-`nil` means visible articles are automatically extended to forward and backward if possible when the commands *N* and *P* (`gnus-Subject-next-article` and `gnus-Subject-prev-article`) are executed in Subject Mode.

`gnus-auto-select-first`
: Non-`nil` means the first unread article is selected automatically when a newsgroup is selected. If you'd like to prevent automatic selection of the first unread article in some newsgroups, set the variable to `nil` in the hook `gnus-Select-group-hook` or `gnus-Apply-kill-hook` (see Section A.5 [Hooks], page 50).

`gnus-auto-select-next`
: Non-`nil` means the next newsgroup is selected automatically at the end of the newsgroup. If the value is `t` and the next newsgroup is empty (no unread articles), GNUS will exit Subject Mode and go back to Group Mode. If the value is neither `nil` nor `t`, GNUS won't exit Subject Mode but will select the following unread newsgroup. If the value is '`quietly`', the next unread newsgroup will be selected without any confirmations.

`gnus-auto-select-same`
: Non-`nil` means an article with the same subject as the current article is selected automatically like '`rn -S`'.

`gnus-auto-center-subject`
: Non-`nil` means that the cursor is always kept centered in the Subject Mode window.

`gnus-break-pages`
: Non-`nil` means an article is broken into pages at page delimiters. The page delimiter is specified by the variable `gnus-page-delimiter`. This may not work with some versions of GNU Emacs earlier than version 18.50.

`gnus-page-delimiter`
: Specifies regexp describing line-beginnings that separate pages of articles. Its default value is `"^\^L"`.

`gnus-digest-show-summary`
: Non-`nil` means that a summary of digest messages is shown when reading a digest article using the command `gnus-Subject-rmail-digest`.

`gnus-digest-separator`
: Specifies a regexp which separates messages in a digest article. Changes to this variable only affect the commands `gnus-Subject-next-digest` and `gnus-Subject-prev-digest`, but not the command `gnus-Subject-rmail-digest`.

`gnus-optional-headers`
: Specifies a function which generates an optional string displayed in the Subject buffer. The function is called with an article *headers*, and must return a string excluding '`[`' and '`]`'. *Headers* is a vector containing headers of the current article. Macros and functions accessing contents of the *headers* are defined as `nntp-header-`*`field`* and `gnus-header-`*`field`*, respectively.

  GNUS provides two functions:

  `gnus-optional-lines-and-from`
  : Return a string like `"`*nnn*:*author*`"`, where *nnn* is the number of lines in an article and *author* is the name of the author.

  `gnus-optional-lines`
  : Return a string like `"`*nnn*`"`, where *nnn* is the number of lines in an article.

  See Section A.5 [Hooks], page 50, to change optional headers according to selected newsgroups.

`gnus-Info-directory`
: Specifies a directory where the GNUS Info file is placed. It is not necessary to change this variable unless you install an Info file in a directory different from the variable `Info-directory`. See Section 1.5 [Texinfo Manual], page 15, for more information.

`gnus-mail-reply-method`
: Specifies a function to begin composing reply mail messages. The function will be called with an optional argument which means yank original article automatically if non-`nil`. To use Mail Mode, set the variable to `gnus-mail-reply-using-mail`. To use mh-e letter Mode, set the variable to `gnus-mail-reply-using-mhe`.

`gnus-mail-forward-method`
: Specifies a function to forward the current message to another user. To use Mail Mode, set the variable to `gnus-mail-forward-using-mail`. To use mh-e letter Mode, set the variable to `gnus-mail-forward-using-mhe`.

`gnus-mail-other-window-method`
: Specifies a function to begin composing mail messages in other window. To use Mail Mode, set the variable to `gnus-mail-other-window-using-mail`. To use mh-e letter Mode, set the variable to `gnus-mail-other-window-using-mhe`.

`gnus-subscribe-newsgroup-method`
: Specifies a function called with a newsgroup name when a new newsgroup is found. GNUS provides the following three functions:

`gnus-subscribe-randomly`
: Inserts a new newsgroup at the beginning of newsgroups. Thus, newsgroups are in random order.

`gnus-subscribe-alphabetically`
: Inserts a new newsgroup in strict alphabetic order.

`gnus-subscribe-hierarchically`
: Inserts a new newsgroup in hierarchical newsgroup order.

The following two definitions illustrate how to write your favorite subscribing method. The following definition (is the definition of the function gnus-subscribe-randomly) adds new newsgroup at the beginning of newsgroups:

```
(setq gnus-subscribe-newsgroup-method
      (function
       (lambda (newsgroup)
         (gnus-subscribe-newsgroup newsgroup
                                   (car (car gnus-newsrc-assoc))))))
```

Instead, if you want to add new newsgroup at the end of newsgroups, use the following:

```
(setq gnus-subscribe-newsgroup-method
      (function
       (lambda (newsgroup)
         (gnus-subscribe-newsgroup newsgroup nil))))
```

If you want to prevent adding new newsgroups automatically and want to subscribe them later using the command *U* (`gnus-Group-unsubscribe-group`) in the Newsgroup buffer, use the following:

```
(setq gnus-subscribe-newsgroup-method
      (function (lambda (newsgroup) nil))) ;Do nothing.
```

The following final example must be the most useful for you who want not to add new newsgroups automatically. This definition subscribes a new newsgroup first, and then kills it. The killed newsgroups can be added to the subscription list interactively using Browse-Killed Mode (see Section 4.3 [Maintenance], page 22).

```
(setq gnus-subscribe-newsgroup-method
      (function
       (lambda (newsgroup)
         (gnus-subscribe-newsgroup newsgroup)
         (gnus-kill-newsgroup newsgroup))))
```

A.2 NNTP Specific Variables

`nntp-large-newsgroup`
: Specifies the number of articles which indicates a large newsgroup. If the number of articles is greater than the value, verbose messages will be shown to indicate the current status.

`nntp-buggy-select`
: Non-`nil` means the select routine of your operating system is buggy. GNUS may hang up while waiting for NNTP server responses. The problem may be solved by setting the variable to `t`. See Section B.1 [NNTP Problems], page 57, for more information.

`nntp-maximum-request`
: Specifies the maximum number of requests to be sent to the NNTP server at one time. GNUS may hang up while retrieving headers of a large newsgroup because sending many requests to the NNTP server without reading replies to them causes deadlock. In this case, set the variable to a lower number. See Section B.1 [NNTP Problems], page 57, for more information.

`nntp-debug-read`
: Non-`nil` means show the communication status about reading the NNTP server output. Set the variable to `nil` if you are annoyed about verbose messages while reading news from slow terminal.

`tcp-program-name`
: Specifies a program which establishes communications between Emacs and the NNTP server. Its default value is `tcp` which is distributed as `tcp.c` with other files of GNUS (see Section 1.1 [Files of GNUS], page 11). If your Emacs has the function `open-network-stream`, there is no need to define this variable.

A.3 Local News Spool Specific Variables

`nnspool-inews-program`
: Specifies a program to post news. This is default to the variable `news-inews-program` which is default to `inews`.

`nnspool-inews-switches`
: Specifies switches for the function `nnspool-request-post` to pass to the command `inews` for posting news. Its default value is `("-h")`.

`nnspool-spool-directory`
: Specifies a directory of a local news spool. This is default to the variable `news-path` which is default to `/usr/spool/news`.

`nnspool-active-file`
: Specifies an active file of the Bnews system for a local news spool. Its default value is `/usr/lib/news/active`.

`nnspool-history-file`
: Specifies a history file of the Bnews system for a local news spool. Its default value is `/usr/lib/news/history`. Some machines may not have this file. In this case, commands to refer to articles by Message-IDs will not work at all (see Section 5.3 [Referencing Articles], page 30).

A.4 Private Directory Specific Variables

`mhspool-list-directory-switches`

Specifies switches for the function `mhspool-request-list` to pass to the command `ls` for getting file listings in a private directory. There should be one entry for each line. Its default value is `("-R")`. Some machines may require the `("-R1")` switch.

A.5 Function Hooks

`gnus-Group-mode-hook`

Called with no arguments after initializing Group Mode if its value is non-`nil`. This hook is intended to customize Group Mode only once. It is possible to define or change the NNTP server as you like in this hook since the hook is called before GNUS is connected to an NNTP server.

`gnus-Subject-mode-hook`

Called with no arguments after initializing Subject Mode if its value is non-`nil`. This hook is intended to customize Subject Mode only once. All sorts of searches in Subject Mode normally ignore the case of the text they are searching through. If you do not want to ignore the case, set the variable `case-fold-search` to `nil` in this hook.

The following example shows how to assign the functions `gnus-Subject-next-group` and `gnus-Subject-prev-group` to keys in Subject Mode.

```
(setq gnus-Subject-mode-hook
      (function
       (lambda ()
         (local-set-key "\C-cn" 'gnus-Subject-next-group)
         (local-set-key "\C-cp" 'gnus-Subject-prev-group))))
```

`gnus-Article-mode-hook`

Called with no arguments after initializing Article Mode if its value is non-`nil`. This hook is intended to customize Article Mode only once.

Displaying the current time in the mode line of buffers is disabled in the Article buffer and the Subject buffer to show information on the current newsgroup and the current article. If you want to display the current time in the mode line of the Article buffer, make the variable `global-mode-string` no longer have a separate value in the buffer as follows:

```
(setq gnus-Article-mode-hook
      (function
       (lambda ()
         (kill-local-variable 'global-mode-string))))
```

`gnus-Kill-file-mode-hook`

Called with no arguments after initializing KILL-File Mode if its value is non-`nil`.

`gnus-Browse-killed-mode-hook`
: Called with no arguments after initializing Browse-Killed Mode if its value is non-`nil`.

`gnus-Open-server-hook`
: Called with no arguments just before opening a connection to NNTP server if its value is non-`nil`.

`gnus-Startup-hook`
: Called with no arguments after an NNTP server is successfully connected to if its value is non-`nil`. It is possible to change the behavior of GNUS according to the server.

`gnus-Group-prepare-hook`
: Called with no arguments after a list of newsgroups is prepared in the Newsgroup buffer. This hook is intended to modify the buffer.

`gnus-Subject-prepare-hook`
: Called with no arguments after list of subjects is prepared in the Subject buffer. This hook is intended to modify the buffer.

`gnus-Article-prepare-hook`
: Called with no arguments after an article is prepared in the Article buffer. This hook is intended to modify the buffer. For example, kanji code conversion or un-ROT13-ing can be done in this hook.

`gnus-Select-group-hook`
: Called with no arguments when a newsgroup is selected. This hook is intended to change the behavior of GNUS according to the selected newsgroup.

  The following is an example of sorting the headers listed in the Subject buffer by date and then by subject. Preceding '`Re:`' of subjects is ignored while comparing subjects.

```
(setq gnus-Select-group-hook
      (function
       (lambda ()
         ;; First of all, sort by date.
         (gnus-sort-headers
          (function
           (lambda (a b)
             (gnus-date-lessp (gnus-header-date a)
                              (gnus-header-date b)))))
         ;; Then sort by subject ignoring `Re:'.
         (gnus-sort-headers
          (function
           (lambda (a b)
             (gnus-string-lessp
              (gnus-simplify-subject
               (gnus-header-subject a) 're-only)
              (gnus-simplify-subject
               (gnus-header-subject b) 're-only)
```

```
                   )))))))
```

The following is an example of simplifying subjects like the `gnus-Subject-next-same-subject` command does:

```
(setq gnus-Select-group-hook
      (function
       (lambda ()
         (mapcar (function
                  (lambda (header)
                    (nntp-set-header-subject
                     header
                     (gnus-simplify-subject
                      (gnus-header-subject header) 're-only))))
                 gnus-newsgroup-headers))))
```

In some newsgroups, author names are meaningless. It is possible to prevent listing author names in the Subject buffer as follows:

```
(setq gnus-Select-group-hook
      (function
       (lambda ()
         (cond ((string-equal "comp.sources.unix"
                              gnus-newsgroup-name)
                (setq gnus-optional-headers
                      (function gnus-optional-lines)))
               (t
                (setq gnus-optional-headers
                      (function
                       gnus-optional-lines-and-from)))))))
```

`gnus-Select-article-hook`

Called with no arguments when an article is selected if its value is non-`nil`.

The default hook definition shows conversation thread subtrees of the selected article automatically as follows:

```
(setq gnus-Select-article-hook
      (function
       (lambda ()
         (gnus-Subject-show-thread))))
```

It is possible to run Rmail on a digest article automatically as follows:

```
(setq gnus-Select-article-hook
      (function
       (lambda ()
         (gnus-Subject-show-thread)
         (cond ((string-equal "comp.sys.sun"
                              gnus-newsgroup-name)
                (gnus-Subject-rmail-digest))
               ((and (string-equal "comp.text"
                                   gnus-newsgroup-name)
                     (string-match "^TeXhax Digest"
                                       (gnus-header-subject
                                        gnus-current-headers)))
                           (gnus-Subject-rmail-digest)
                           )))))
```

`gnus-Select-digest-hook`

Called with no arguments when reading digest messages using Rmail if its value is non-`nil`. This hook is intended to modify an article so that Rmail can work with it. See Section 5.1.6 [Digest Articles], page 29, for more information on digest articles.

The following example is the default hook definition to modify incomplete digest articles:

```
(setq gnus-Select-digest-hook
      (function
       (lambda ()
         ;; Reply-To: is required by
         ;; `undigestify-rmail-message'.
         (or (mail-position-on-field "Reply-to" t)
             (progn
               (mail-position-on-field "Reply-to")
               (insert (gnus-fetch-field "From")))))))
```

`gnus-Rmail-digest-hook`

Called with no arguments when reading digest messages using Rmail if its value is non-`nil`. This hook is intended to customize Rmail Mode for reading digest articles.

`gnus-Apply-kill-hook`

Called with no arguments when a newsgroup is selected and the Subject buffer is prepared if its value is non-`nil`. This hook is intended to apply KILL files to the selected newsgroup. It is set to the function `gnus-apply-kill-file` by default.

Since a general KILL file is too heavy to use only for a few newsgroups, a lighter hook function is recommended. For example, if you'd like to apply kills to articles which contain the string '`rmgroup`' in subject in newsgroup '`control`', you can use the following hook:

```
(setq gnus-Apply-kill-hook
      (function
       (lambda ()
         (cond ((string-match "control" gnus-newsgroup-name)
                (gnus-kill "Subject" "rmgroup")
                (gnus-expunge "X"))))))
```

See Chapter 7 [KILL File], page 37, for more information on KILL files.

`gnus-Mark-article-hook`

Called with no arguments when an article is selected for the first time if its value is non-`nil`. The hook is intended to mark an article as read (or unread) automatically when it is selected.

The following example is the default definition of the hook:

```
(setq gnus-Mark-article-hook
      (function
       (lambda ()
         ;; Mark the selected article as read.
         (or (memq gnus-current-article gnus-newsgroup-marked)
             (gnus-Subject-mark-as-read gnus-current-article))
         ;; Put "+" on the current subject.
         (gnus-Subject-set-current-mark "+"))))
```

It is possible to mark as unread ('-') instead when an article is selected as follows:

```
(setq gnus-Mark-article-hook
      (function
       (lambda ()
         ;; Mark the selected article as unread.
         (gnus-Subject-mark-as-unread gnus-current-article)
         ;; Put "+" on the current subject.
         (gnus-Subject-set-current-mark "+"))))
```

`gnus-Inews-article-hook`
: Called with no arguments before posting an article if its value is non-`nil`. This hook is called just before sending an article to the NNTP server or calling the `inews` program, while the hook `news-inews-hook` is called before preparing article headers. This hook is intended to run special encoding programs such as kanji code conversion on the article.

`gnus-Exit-group-hook`
: Called with no arguments when exiting the current newsgroup if its value is non-`nil`. If your machine is so slow that exiting from Subject Mode takes a long time, you can inhibit marking articles as read by using cross-reference information in the '`Xref:`' field by setting the variable `gnus-newsgroup-headers` to `nil` in this hook.

`gnus-Exit-gnus-hook`
: Called with no arguments when exiting GNUS if its value is non-`nil`. If you want to clear out Emacs buffers which were created by GNUS and remain afterwards, you can use this hook.

The following example shows how to kill a buffer which was used for posting news.

```
(setq gnus-Exit-gnus-hook
      (function
       (lambda ()
         ;; Kill a buffer used for posting news.
         (and (get-buffer "*post-news*")
              (kill-buffer "*post-news*")))))
```

`gnus-Suspend-gnus-hook`
: Called with no arguments when suspending GNUS if its value is non-`nil`. The purpose is the same as the hook `gnus-Exit-gnus-hook`.

`gnus-Save-newsrc-hook`
: Called with no arguments before saving the startup file `.newsrc` if its value is non-`nil`. This hook is intended to change the way of backing up the startup file.

`nntp-server-hook`
: Called with no arguments when the connection between Emacs and the NNTP server is established if its value is non-`nil`. This hook is intended to change the kanji code of a buffer associated with the stream. Use the variable `nntp-server-name` to refer to the name of the NNTP server in this hook. See Section B.2 [Kanji Handling], page 57, for more information.

Appendix B Troubleshooting

Some common problems and their solutions are described. If you have any other problems which are not described here and cannot solve them by yourself, see Appendix C [Reporting Bugs], page 61.

B.1 NNTP Problems

B.1.1 Infinite Loop Caused by Buggy Select Routine

Emacs may hang up while waiting for NNTP server responses. This may be caused by a buggy select routine of your operating system. If so, the problem may be solved by using source codes of `nntp.el` instead of byte-compiled codes. If you still have problems with it, set the variable `nntp-buggy-select` to `t`.

B.1.2 Deadlock Caused by Packet Overflow

Emacs may hang up while retrieving headers of a large newsgroup. The reason is that too many requests have been sent to the NNTP server without reading replies to them. This causes a deadlock of Emacs and the server. In this case, the number of requests sent to the server at one time must be reduced. Set the variable `nntp-maximum-request` to a lower value than the default. The optimal value depends on your computing environment.

B.2 Kanji Handling

B.2.1 Kanji Handling In NEmacs 2.1

If the kanji code of articles stored in the NNTP server is different from your private file kanji code, the correct kanji code of the buffer associated with the NNTP stream must be specified using the hook `nntp-server-hook` as follows:

```
(setq nntp-server-hook
      (function
       (lambda ()
         ;; Server's Kanji code is EUC (NEmacs hack).
         (make-local-variable 'kanji-fileio-code)
         (setq kanji-fileio-code 0))))
```

If you use a local news spool in stead of NNTP, the following additional hook is required to post an article in the correct kanji code in any case:

```
(setq gnus-Inews-article-hook
      (function
       (lambda ()
         (save-excursion
           (set-buffer nntp-server-buffer)
           (make-local-variable 'kanji-fileio-code)
           (setq kanji-fileio-code 0)   ;EUC
           ))))
```

The variable `nntp-server-name` is a buffer local variable holding a host name running an NNTP server. Use this variable to change the kanji code according to the server. The following example shows how to change the kanji code using this variable.

```
(setq nntp-server-hook
      (function
       (lambda ()
         (make-local-variable 'kanji-fileio-code)
         (cond ((string-equal nntp-server-name "foo")
                (setq kanji-fileio-code 0)) ;EUC
               ((string-equal nntp-server-name "bar")
                (setq kanji-fileio-code 1)) ;Shift-JIS
               (t
                (setq kanji-fileio-code 2)) ;JIS
               ))))
```

B.2.2 Kanji Handling In NEmacs 3.0

The author has less experiences with NEmacs 3.0. So, the following description may be wrong. If you find any mistakes, please let the author know (see Appendix C [Reporting Bugs], page 61).

In NEmacs 3.0, the kanji code of articles stored in the NNTP server must be specified according to both the service name (or number) and the host name running the server. For this, the function `define-service-kanji-code` is provided.

The following example specifies the kanji code of the host '`flab`' as JIS using this function:

```
(define-service-kanji-code "nntp" "flab" 2)      ;2 stands for JIS.
```

B.2.3 Kanji Handling In SX/A Emacs

If the kanji code of articles stored in the NNTP server is not EUC, it must be converted to EUC in an Emacs buffer after being read into the buffer. The kanji code of articles being posted must be also converted to the server specific kanji code in an Emacs buffer before actually being sent to the server. The following examples show how to do these using hooks:

```
(setq gnus-Article-prepare-hook
      (function
       (lambda ()
         (call-process-region (point-min) (point-max)
                              "nkf" t t nil "-e" ;-e stands for EUC.
                              ))))
(setq gnus-Inews-article-hook
      (function
       (lambda ()
         (call-process-region (point-min) (point-max)
                              "nkf" t t nil "-j" ;-j stands for JIS.
                              ))))
```

In this example, `nkf` (Network Kanji Filter) is used as a kanji code converter, and the kanji code of the NNTP server is JIS.

B.3 Preloading GNUS

Basically, GNUS is not designed to be preloaded. For instance, if you preload `gnus.el`, some user variables which are initialized from environment variables may be improperly initialized according to your environment definitions. The variables `gnus-nntp-server`, `gnus-author-copy`, and `gnus-article-save-directory` are such variables.

To prevent GNUS from being initialized from your definitions, you should `unsetenv` related environment variables before preloading GNUS, or set the variables to `nil` after loading GNUS.

Appendix C Reporting Bugs

C.1 Mailing Lists and USENET Newsgroup

There are two mailing lists and one USENET newsgroup for discussing GNUS related topics. These are intended for exchanging useful information about GNUS, such as bug reports, useful hooks, and extensions of GNUS. If you have any questions or problems, feel free to ask about them. Suggestions are also welcome.

`gnu.emacs.gnus`
: This is a USENET newsgroup under the gnu.all hierarchy which is concerned with the GNU Project of the Free Software Foundation.

`info-gnus-english@tut.cis.ohio-state.edu`
: This is an Internet mailing list which is gated bi-directionally with the gnu.emacs.gnus newsgroup. English is the official language of the list. Please send subscription requests to:

  info-gnus-english-request@tut.cis.ohio-state.edu

`info-gnus@flab.Fujitsu.CO.JP`
: This is a JUNET mailing list. Messages of info-gnus-english and gnu.emacs.gnus are forwarded to this list. English and Japanese are the official languages of the list. Please send subscription requests to:

  info-gnus-request@flab.Fujitsu.CO.JP

The major difference between info-gnus-english/gnu.emacs.gnus and info-gnus is the official language. There is no need to subscribe to info-gnus if you cannot read messages written in Japanese since most discussions and important announcements will be sent to info-gnus-english.

C.2 How to Report a Bug

If you find a bug, it is important to report it and to report it in a way which is useful. If it is a bug of a lisp program, what is the most useful is an exact backtrace information of the lisp program with the version number of GNUS that you are using.

To make the backtrace information, you must set the Emacs variable `debug-on-error` to `t` before the error happens. A backtrace obtained from a byte-compiled lisp program is not usually understandable. To make a human readable backtrace, load the source program which is not byte-compiled yet and then produce the error.

See Info file `emacs`, node '`Bugs`', for more information.

Key (Character) Index

Command and Function Index

Variable Index

Program Index

Concept Index

Short Contents

Table of Contents