GNU Emacs FAQ

[This version has been somewhat edited from the last-posted version (as of August 1999) for inclusion in the Emacs distribution.]

This is the GNU Emacs FAQ, last updated on 9 February 2005.

The FAQ is maintained as a Texinfo document, allowing us to create HTML, Info, and TeX documents from a single source file, and is slowly but surely being improved. Please bear with us as we improve on this format. If you have any suggestions or questions, please contact the FAQ maintainers (`emacs-faq@lerner.co.il`).

1 FAQ notation

This chapter describes notation used in the GNU Emacs FAQ, as well as in the Emacs documentation. Consult this section if this is the first time you are reading the FAQ, or if you are confused by notation or terms used in the FAQ.

1.1 What do these mean: *C-h*, *M-C-*`a`, ⟨RET⟩, ⟨ESC⟩ `a`, etc.?

- *C-x*: press the ⟨x⟩ key while holding down the ⟨Control⟩ key
- *M-x*: press the ⟨x⟩ key while holding down the ⟨Meta⟩ key (if your computer doesn't have a ⟨Meta⟩ key, see Section 10.13 [No Meta key], page 64)
- *M-C-x*: press the ⟨x⟩ key while holding down both ⟨Control⟩ and ⟨Meta⟩
- *C-M-x*: a synonym for the above
- ⟨LFD⟩: Linefeed or Newline; same as *C-j*
- ⟨RET⟩: ⟨Return⟩, sometimes marked ⟨Enter⟩; same as *C-m*
- ⟨DEL⟩: ⟨Delete⟩, usually **not** the same as ⟨Backspace⟩; same as *C-?* (see Section 10.9 [Backspace invokes help], page 62, if deleting invokes Emacs help)
- ⟨ESC⟩: Escape; same as *C-[*
- ⟨TAB⟩: Tab; same as *C-i*
- ⟨SPC⟩: Space bar

Key sequences longer than one key (and some single-key sequences) are written inside quotes or on lines by themselves, like this:

```
M-x frobnicate-while-foo RET
```

Any real spaces in such a key sequence should be ignored; only ⟨SPC⟩ really means press the space key.

The ASCII code sent by *C-x* (except for *C-?*) is the value that would be sent by pressing just ⟨x⟩ minus 96 (or 64 for upper-case ⟨X⟩) and will be from 0 to 31. On Unix and GNU/Linux terminals, the ASCII code sent by *M-x* is the sum of 128 and the ASCII code that would be sent by pressing just ⟨x⟩. Essentially, ⟨Control⟩ turns off bits 5 and 6 and ⟨Meta⟩ turns on bit 7[1].

C-? (aka ⟨DEL⟩) is ASCII code 127. It is a misnomer to call *C-?* a "control" key, since 127 has both bits 5 and 6 turned ON. Also, on very few keyboards does *C-?* generate ASCII code 127.

See Info file '`emacs`', node '`Text Characters`', and See Info file '`emacs`', node '`Keys`', for more information. (See Section 1.3 [On-line manual], page 4, for more information about Info.)

[1] DOS and Windows terminals don't set bit 7 when the ⟨Meta⟩ key is pressed.

1.2 What does 'M-x *command*' mean?

M-x command means type *M-x*, then type the name of the command, then type ⟨RET⟩. (See Section 1.1 [Basic keys], page 3, if you're not sure what *M-x* and ⟨RET⟩ mean.)

M-x (by default) invokes the command `execute-extended-command`. This command allows you to run any Emacs command if you can remember the command's name. If you can't remember the command's name, you can type ⟨TAB⟩ and ⟨SPC⟩ for completion, ⟨?⟩ for a list of possibilities, and *M-p* and *M-n* (or up-arrow and down-arrow on terminals that have these editing keys) to see previous commands entered. An Emacs *command* is an *interactive* Emacs function.

Your system administrator may have bound other key sequences to invoke `execute-extended-command`. A function key labeled *Do* is a good candidate for this, on keyboards that have such a key.

If you need to run non-interactive Emacs functions, see Section 5.29 [Evaluating Emacs Lisp code], page 29.

1.3 How do I read topic XXX in the on-line manual?

When we refer you to some *topic* in the on-line manual, you can read this manual node inside Emacs (assuming nothing is broken) by typing *C-h i m emacs* ⟨RET⟩ *m topic* ⟨RET⟩.

This invokes Info, the GNU hypertext documentation browser. If you don't already know how to use Info, type ⟨?⟩ from within Info.

If we refer to *topic*:*subtopic*, type *C-h i m emacs* ⟨RET⟩ *m topic* ⟨RET⟩ *m subtopic* ⟨RET⟩.

If these commands don't work as expected, your system administrator may not have installed the Info files, or may have installed them improperly. In this case you should complain.

See Section 3.3 [Getting a printed manual], page 12, if you would like a paper copy of the Emacs manual.

1.4 What are 'etc/SERVICE', 'src/config.h', and 'lisp/default.el'?

These are files that come with Emacs. The Emacs distribution is divided into subdirectories; the important ones are '`etc`', '`lisp`', and '`src`'.

If you use Emacs, but don't know where it is kept on your system, start Emacs, then type *C-h v data-directory* ⟨RET⟩. The directory name displayed by this will be the full pathname of the installed '`etc`' directory. (This full path is recorded in the Emacs variable `data-directory`, and *C-h v* displays the value and the documentation of a variable.)

The location of your Info directory (i.e., where on-line documentation is stored) is kept in the variable `Info-default-directory-list`. Use *C-h v Info-default-directory-list* ⟨RET⟩ to see the value of this variable, which will be a list of directory names. The last directory in that list is probably where most Info files are stored. By default, Info documentation is placed in '`/usr/local/info`'.

Some of these files are available individually via FTP or e-mail; see Section 3.8 [Informational files for Emacs], page 14. They all are available in the source distribution. Many of the files in the '`etc`' directory are also available via the Emacs '`Help`' menu, or by typing *`C-h ?`* (*`M-x help-for-help`*).

Your system administrator may have removed the '`src`' directory and many files from the '`etc`' directory.

1.5 What are FSF, LPF, OSF, GNU, RMS, FTP, and GPL?

FSF Free Software Foundation

LPF League for Programming Freedom

OSF Open Software Foundation

GNU GNU's Not Unix

RMS Richard Matthew Stallman

FTP File Transfer Protocol

GPL GNU General Public License

Avoid confusing the FSF, the LPF, and the OSF. The LPF opposes look-and-feel copyrights and software patents. The FSF aims to make high quality free software available for everyone. The OSF is a consortium of computer vendors which develops commercial software for Unix systems.

The word "free" in the title of the Free Software Foundation refers to "freedom," not "zero dollars." Anyone can charge any price for GPL-covered software that they want to. However, in practice, the freedom enforced by the GPL leads to low prices, because you can always get the software for less money from someone else, since everyone has the right to resell or give away GPL-covered software.

2 General questions

This chapter contains general questions having to do with Emacs, the Free Software Foundation, and related organizations.

2.1 What is the LPF?

The LPF opposes the expanding danger of software patents and look-and-feel copyrights. To get more information, feel free to contact the LPF via e-mail or otherwise. You may also contact Joe Wells (`jbw@cs.bu.edu`); he will be happy to talk to you about the LPF.

You can find more information about the LPF in the file '`etc/LPF`'. More papers describing the LPF's views are available on the Internet and also from the LPF home page (`http://lpf.ai.mit.edu/`).

2.2 What is the real legal meaning of the GNU copyleft?

The real legal meaning of the GNU General Public License (copyleft) will only be known if and when a judge rules on its validity and scope. There has never been a copyright infringement case involving the GPL to set any precedents. Please take any discussion regarding this issue to the newsgroup `news:gnu.misc.discuss`, which was created to hold the extensive flame wars on the subject.

RMS writes:

> The legal meaning of the GNU copyleft is less important than the spirit, which is that Emacs is a free software project and that work pertaining to Emacs should also be free software. "Free" means that all users have the freedom to study, share, change and improve Emacs. To make sure everyone has this freedom, pass along source code when you distribute any version of Emacs or a related program, and give the recipients the same freedom that you enjoyed.

2.3 What are appropriate messages for `news:gnu.emacs.help`, `news:gnu.emacs.bug`, `news:comp.emacs`, etc.?

The file '`etc/MAILINGLISTS`' describes the purpose of each GNU mailing list. (See Section 3.8 [Informational files for Emacs], page 14, if you want a copy of the file.) For those lists which are gatewayed with newsgroups, it lists both the newsgroup name and the mailing list address.

The newsgroup `news:comp.emacs` is for discussion of Emacs programs in general. This includes Emacs along with various other implementations, such as XEmacs, JOVE, MicroEmacs, Freemacs, MG, Unipress, CCA, and Epsilon.

Many people post Emacs questions to `news:comp.emacs` because they don't receive any of the `gnu.*` newsgroups. Arguments have been made both for and against posting GNU-Emacs-specific material to `news:comp.emacs`. You have to decide for yourself.

Messages advocating "non-free" software are considered unacceptable on any of the `gnu.*` newsgroups except for `news:gnu.misc.discuss`, which was created to hold the extensive flame-wars on the subject. "Non-free" software includes any software for which the end user can't freely modify the source code and exchange enhancements. Be careful to remove the `gnu.*` groups from the '`Newsgroups:`' line when posting a followup that recommends such software.

`news:gnu.emacs.bug` is a place where bug reports appear, but avoid posting bug reports to this newsgroup directly (see Section 2.5 [Reporting bugs], page 8).

2.4 Where can I get old postings to `news:gnu.emacs.help` and other GNU groups?

The FSF has maintained archives of all of the GNU mailing lists for many years, although there may be some unintentional gaps in coverage. The archive is not particularly well organized or easy to retrieve individual postings from, but pretty much everything is there.

The archive is at `ftp://ftp-mailing-list-archives.gnu.org`.

As of this writing, the archives are not yet working.

Web-based Usenet search services, such as DejaNews (`http://www.dejanews.com`), also archive the `gnu.*` groups.

2.5 Where should I report bugs and other problems with Emacs?

The correct way to report Emacs bugs is by e-mail to `bug-gnu-emacs@gnu.org`. Anything sent here also appears in the newsgroup `news:gnu.emacs.bug`, but please use e-mail instead of news to submit the bug report. This ensures a reliable return address so you can be contacted for further details.

Be sure to read the "Bugs" section of the Emacs manual before reporting a bug to bug-gnu-emacs! The manual describes in detail how to submit a useful bug report. (See Section 1.3 [On-line manual], page 4, if you don't know how to read the manual.)

RMS says:

> Sending bug reports to `help-gnu-emacs@gnu.org` (which has the effect of posting on `news:gnu.emacs.help`) is undesirable because it takes the time of an unnecessarily large group of people, most of whom are just users and have no idea how to fix these problem. `bug-gnu-emacs@gnu.org` reaches a much smaller group of people who are more likely to know what to do and have expressed a wish to receive more messages about Emacs than the others.

RMS says it is sometimes fine to post to `news:gnu.emacs.help`:

> If you have reported a bug and you don't hear about a possible fix, then after a suitable delay (such as a week) it is okay to post on `gnu.emacs.help` asking if anyone can help you.

If you are unsure whether you have found a bug, consider the following non-exhaustive list, courtesy of RMS:

> If Emacs crashes, that is a bug. If Emacs gets compilation errors while building, that is a bug. If Emacs crashes while building, that is a bug. If Lisp code does not do what the documentation says it does, that is a bug.

2.6 How do I unsubscribe from this mailing list?

If you are receiving a GNU mailing list named *list*, you might be able to unsubscribe from it by sending a request to the address *list*`-request@gnu.org`. However, this will not work if you are not listed on the main mailing list, but instead receive the mail from a distribution point. In that case, you will have to track down at which distribution point you are listed. Inspecting the '`Received`' headers on the mail messages may help, along with liberal use of the '`EXPN`' or '`VRFY`' sendmail commands through '`telnet` *site-address* `smtp`'. Ask your postmaster for help, if you cannot figure out these details.

2.7 What is the current address of the FSF?

E-mail
: gnu@gnu.org

Telephone
: +1-617-542-5942

Fax
: +1-617-542-2652

World Wide Web
: `http://www.gnu.org/`

Postal address
: Free Software Foundation
59 Temple Place - Suite 330
Boston, MA 02111-1307
USA

For details on how to order items directly from the FSF, see the GNU Web site (`http://www.gnu.org/order/order.html`), and also the files '`etc/ORDERS`', '`ORDERS.EUROPE`', and '`ORDERS.JAPAN`'.

3 Getting help

This chapter tells you how to get help with Emacs

3.1 I'm just starting Emacs; how do I do basic editing?

Type *C-h t* to invoke the self-paced tutorial. Just typing *C-h* enters the help system.

Your system administrator may have changed *C-h* to act like ⟨DEL⟩ to deal with local keyboards. You can use *M-x help-for-help* instead to invoke help. To discover what key (if any) invokes help on your system, type *M-x where-is* ⟨RET⟩ *help-for-help* ⟨RET⟩. This will print a comma-separated list of key sequences in the echo area. Ignore the last character in each key sequence listed. Each of the resulting key sequences invokes help.

Emacs help works best if it is invoked by a single key whose value should be stored in the variable `help-char`.

There is also a WWW-based tutorial for Emacs 18, much of which is also relevant for later versions of Emacs, available at

`http://kufacts.cc.ukans.edu/cwis/writeups/misc/emacsguide.html`

3.2 How do I find out how to do something in Emacs?

There are several methods for finding out how to do things in Emacs.

- The complete text of the Emacs manual is available on-line via the Info hypertext reader. Type *C-h i* to invoke Info. Typing ⟨h⟩ immediately after entering Info will provide a short tutorial on how to use it.
- To quickly locate the section of the manual which discusses a certain issue, or describes a command or a variable, type *C-h i m emacs* ⟨RET⟩ *i topic* ⟨RET⟩, where *topic* is the name of the topic, the command, or the variable which you are looking for. If this does not land you on the right place in the manual, press *,* (comma) repeatedly until you find what you need. (The *i* and *,* keys invoke the index-searching functions, which look for the *topic* you type in all the indices of the Emacs manual.)
- You can list all of the commands whose names contain a certain word (actually which match a regular expression) using *C-h a* (*M-x command-apropos*).
- The command *C-h C-f* (`Info-goto-emacs-command-node`) prompts for the name of a command, and then attempts to find the section in the Emacs manual where that command is described.
- You can list all of the functions and variables whose names contain a certain word using *M-x apropos*.
- You can list all of the functions and variables whose documentation matches a regular expression or a string, using *M-x apropos-documentation*.
- You can order a hardcopy of the manual from the FSF. See Section 3.3 [Getting a printed manual], page 12.
- You can get a printed reference card listing commands and keys to invoke them. You can order one from the FSF for $1 (or 10 for $5), or you can print your own from the

'etc/refcard.tex' or 'etc/refcard.ps' files in the Emacs distribution. Beginning with version 21.1, the Emacs distribution comes with translations of the reference card into several languages; look for files named 'etc/*lang*-refcard.*', where *lang* is a two-letter code of the language. For example, the German version of the reference card is in the files 'etc/de-refcard.tex' and 'etc/de-refcard.ps'.

- There are many other commands in Emacs for getting help and information. To get a list of these commands, type '?' after *C-h*.

3.3 How do I get a printed copy of the Emacs manual?

You can order a printed copy of the Emacs manual from the FSF. For details see the GNU Web site (http://www.gnu.org/order/order.html) and the file 'etc/ORDERS'.

The full Texinfo source for the manual also comes in the 'man' directory of the Emacs distribution, if you're daring enough to try to print out this 620-page manual yourself (see Section 3.6 [Printing a Texinfo file], page 14).

If you absolutely have to print your own copy, and you don't have TEX, you can get a PostScript version from

http://www.gnu.org/manual/emacs/ps/emacs.ps.gz

An HTML version of the manual is at

www.gnu.org/manual/emacs/index.html

See Section 3.2 [Learning how to do something], page 11, for how to view the manual on-line.

3.4 Where can I get documentation on Emacs Lisp?

Within Emacs, you can type *C-h f* to get the documentation for a function, *C-h v* for a variable.

For more information, obtain the Emacs Lisp Reference Manual. Details on ordering it from FSF are on the GNU Web site (http://www.gnu.org/order/order.html) and in the file 'etc/ORDERS'.

The Emacs Lisp Reference Manual is also available on-line, in Info format. Texinfo source for the manual (along with pregenerated Info files) is available at

ftp://ftp.gnu.org/pub/gnu/emacs/elisp-manual-21-2.6.tar.gz

and all mirrors of 'ftp.gnu.org' (for a list, see Section 8.5 [Current GNU distributions], page 50). See Section 3.5 [Installing Texinfo documentation], page 13, if you want to install the Info files, or Section 3.6 [Printing a Texinfo file], page 14, if you want to use the Texinfo source to print the manual yourself.

An HTML version of the Emacs Lisp Reference Manual is available at

http://www.gnu.org/manual/elisp-manual-21-2.6/elisp.html

3.5 How do I install a piece of Texinfo documentation?

First, you must turn the Texinfo files into Info files. You may do this using the stand-alone '`makeinfo`' program, available as part of the latest Texinfo package at

```
ftp://ftp.gnu.org/pub/gnu/texinfo/texinfo-4.0.tar.gz
```

and all mirrors of '`ftp.gnu.org`' (for a list, see Section 8.5 [Current GNU distributions], page 50).

For information about the Texinfo format, read the Texinfo manual which comes with the Texinfo package. This manual also comes installed in Info format, so you can read it on-line; type *`C-h i m texinfo`* ⟨RET⟩.

Alternatively, you could use the Emacs command *`M-x texinfo-format-buffer`*, after visiting the Texinfo source file of the manual you want to convert.

Neither `texinfo-format-buffer` nor '`makeinfo`' installs the resulting Info files in Emacs's Info tree. To install Info files, perform these steps:

1. Move the files to the '`info`' directory in the installed Emacs distribution. See Section 1.4 [File-name conventions], page 4, if you don't know where that is.
2. Run the `install-info` command, which is part of the Texinfo distribution, to update the main Info directory menu, like this:

   `install-info --info-dir=`*dir-path dir-path*`/`*file*

   where *dir-path* is the full path to the directory where you copied the produced Info file(s), and *file* is the name of the Info file you produced and want to install.

   If you don't have the `install-info` command installed, you can edit the file '`info/dir`' in the installed Emacs distribution, and add a line for the top level node in the Info package that you are installing. Follow the examples already in this file. The format is:

   ```
   * Topic: (relative-pathname).  Short description of topic.
   ```

If you want to install Info files and you don't have the necessary privileges, you have several options:

- Info files don't actually need to be installed before being used. You can feed a file name to the `Info-goto-node` command (invoked by pressing ⟨g⟩ in Info mode) by typing the name of the file in parentheses. This goes to the node named "Top" in that file. For example, to view a Info file named '*info-file*' in your home directory, you can type this:

  `C-h i g (~/info-file)` ⟨RET⟩
- You can create your own Info directory. You can tell Emacs where that Info directory is by adding its pathname to the value of the variable `Info-default-directory-list`. For example, to use a private Info directory which is a subdirectory of your home directory named '`Info`', you could put this in your '`.emacs`' file:

  ```
  (setq Info-default-directory-list
        (cons "~/Info" Info-default-directory-list))
  ```

  You will need a top-level Info file named '`dir`' in this directory which has everything the system '`dir`' file has in it, except it should list only entries for Info files in that directory. You might not need it if all files in this directory were referenced by other '`dir`' files. The node lists from all '`dir`' files in `Info-default-directory-list` are merged by the Info system.

3.6 How do I print a Texinfo file?

You can't get nicely printed output from Info files; you must still have the original Texinfo source file for the manual you want to print.

Assuming you have TEX installed on your system, follow these steps:

1. Make sure the first line of the Texinfo file looks like this:

   ```
   \input texinfo
   ```

   You may need to change '`texinfo`' to the full pathname of the '`texinfo.tex`' file, which comes with Emacs as '`man/texinfo.tex`' (or copy or link it into the current directory).
2. Type `texi2dvi` *texinfo-source*, where *texinfo-source* is the name of the Texinfo source file for which you want to produce a printed copy.

   The '`texi2dvi`' script is part of the GNU Texinfo distribution (see Section 3.5 [Installing Texinfo documentation], page 13).
3. Print the DVI file '*texinfo-source*`.dvi`' in the normal way for printing DVI files at your site. For example, if you have a PostScript printer, run the `dvips` program to print the DVI file on that printer.

To get more general instructions, retrieve the latest Texinfo package (see Section 3.5 [Installing Texinfo documentation], page 13).

3.7 Can I view Info files without using Emacs?

Yes. Here are some alternative programs:

- `info`, a stand-alone version of the Info program, comes as part of the Texinfo package. See Section 3.5 [Installing Texinfo documentation], page 13, for details.
- Xinfo, a stand-alone version of the Info program that runs under X Window system. You can get it at `ftp://ftp.gnu.org/pub/gnu/xinfo/xinfo-1.01.01.tar.gz` and all mirrors of '`ftp.gnu.org`' (see Section 8.5 [Current GNU distributions], page 50, for a list of mirrors).
- Tkinfo, an Info viewer that runs under X Window system and uses Tcl/Tk. You can get Tkinfo at `http://math-www.uni-paderborn.de/~axel/tkinfo/`.

3.8 What informational files are available for Emacs?

This isn't a frequently asked question, but it should be! A variety of informational files about Emacs and relevant aspects of the GNU project are available for you to read.

The following files are available in the '`etc`' directory of the Emacs distribution (see Section 1.4 [File-name conventions], page 4, if you're not sure where that is).

'`COPYING`' Emacs General Public License

'`DISTRIB`' Emacs Availability Information, including the popular "Free Software Foundation Order Form"

'`FTP`' How to get GNU Software by Internet FTP or by UUCP

'GNU' The GNU Manifesto

'INTERVIEW'
Richard Stallman discusses his public-domain UNIX-compatible software system with BYTE editors

'LPF' Why you should join the League for Programming Freedom

'MACHINES'
Status of Emacs on Various Machines and Systems

'MAILINGLISTS'
GNU Project Electronic Mailing Lists

'NEWS' Emacs news, a history of recent user-visible changes

'SERVICE' GNU Service Directory

'SUN-SUPPORT'
including "Using Emacstool with GNU Emacs"

Latest versions of the above files also available at

`ftp://ftp.gnu.org/pub/gnu/GNUinfo/`

More GNU information, including back issues of the *GNU's Bulletin*, are at

`http://www.gnu.org/bulletins/bulletins.html` and

`http://www.cs.pdx.edu/~trent/gnu/gnu.html`

3.9 Where can I get help in installing Emacs?

See Section 7.1 [Installing Emacs], page 45, for some basic installation hints, and see Section 7.3 [Problems building Emacs], page 46, or Section 7.4 [Linking with -lX11 fails], page 47, if you have problems with the installation.

The file 'etc/SERVICE' (see Section 1.4 [File-name conventions], page 4, if you're not sure where that is) lists companies and individuals willing to sell you help in installing or using Emacs. An up-to-date version this file is available on 'ftp.gnu.org' (see Section 3.8 [Informational files for Emacs], page 14).

3.10 Where can I get the latest version of this FAQ?

The Emacs FAQ is available in several ways:

- Inside of Emacs itself. You can get it from selecting the 'Emacs FAQ' option from the 'Help' menu of the Emacs menu bar at the top of any Emacs frame, or by typing *C-h F* (*M-x view-emacs-FAQ*).
- Via USENET. If you can read news, the FAQ should be available in your news spool, in both the `news:gnu.emacs.help` and `news:comp.emacs` newsgroups. Every news reader should allow you to read any news article that is still in the news spool, even if you have read the article before. You may need to read the instructions for your news reader to discover how to do this. In 'rn', this command will do this for you at the article selection level:

```
?GNU Emacs Frequently Asked Questions?rc:m
```

In Gnus, you should type *C-u C-x C-s* from the '`*Summary*`' buffer or *C-u* ⟨SPC⟩ from the '`*Newsgroup*`' buffer to view all articles in a newsgroup.

If the FAQ articles have expired and have been deleted from your news spool, it might (or might not) do some good to complain to your news administrator, because the most recent FAQ should not expire for a while.

- Via HTTP or FTP. You can always fetch the latest FAQ from

  `http://www.lerner.co.il/emacs/` and

  `ftp://ftp.lerner.co.il/pub/emacs/`
- In the Emacs distribution. Since Emacs 18.56, the FAQ at the time of release has been part of the Emacs distribution as either '`etc/FAQ`' or '`man/faq.texi`' (see Section 1.4 [File-name conventions], page 4).
- Via the World Wide Web. A hypertext version is available at

  `http://www.lerner.co.il/emacs/`
- Via anonymous ftp and e-mail from '`rtfm.mit.edu`' (and its mirror in Europe), the main repository for FAQs and other items posted to news.answers. The Emacs FAQs are available at

  `ftp://rtfm.mit.edu/pub/usenet/comp.emacs/` and

  `ftp://ftp.uni-paderborn.de/pub/doc/FAQ/comp/emacs/`

  If you do not have access to anonymous FTP, you can access the archives using the '`rtfm.mit.edu`' mail server. The Emacs FAQ can be retrieved by sending mail to `mail-server@rtfm.mit.edu` with a blank subject and containing

  ```
  send usenet/news.answers/GNU-Emacs-FAQ/diffs
  send usenet/news.answers/GNU-Emacs-FAQ/part1
  send usenet/news.answers/GNU-Emacs-FAQ/part2
  send usenet/news.answers/GNU-Emacs-FAQ/part3
  send usenet/news.answers/GNU-Emacs-FAQ/part4
  send usenet/news.answers/GNU-Emacs-FAQ/part5
  ```

  For more information, send email to `mail-server@rtfm.mit.edu` with "help" and "index" in the body on separate lines.
- As the very last resort, you can e-mail a request to `emacs-faq@lerner.co.il`. Don't do this unless you have made a good-faith effort to obtain the FAQ list via one of the methods listed above.

4 Status of Emacs

This chapter gives you basic information about Emacs, including its latest version status.

4.1 Where does the name “Emacs” come from?

Emacs originally was an acronym for Editor MACroS. RMS says he “picked the name Emacs because ⟨E⟩ was not in use as an abbreviation on ITS at the time.” The first Emacs was a set of macros written in 1976 at MIT by RMS for the editor TECO (Text Editor and COrrector, originally Tape Editor and COrrector) under ITS on a PDP-10. RMS had already extended TECO with a “real-time” full-screen mode with reprogrammable keys. Emacs was started by Guy Steele (`gls@east.sun.com`) as a project to unify the many divergent TECO command sets and key bindings at MIT, and completed by RMS.

Many people have said that TECO code looks a lot like line noise; you can read more at `news:alt.lang.teco`. Someone has written a TECO implementation in Emacs Lisp (to find it, see Section 8.3 [Packages that do not come with Emacs], page 50); it would be an interesting project to run the original TECO Emacs inside of Emacs.

For some not-so-serious alternative reasons for Emacs to have that name, check out the file ‘`etc/JOKES`’ (see Section 1.4 [File-name conventions], page 4).

4.2 What is the latest version of Emacs?

Emacs 21.2 is the current version as of this writing.

4.3 What is different about Emacs 20?

To find out what has changed in recent versions, type *C-h n* (*M-x view-emacs-news*). The oldest changes are at the bottom of the file, so you might want to read it starting there, rather than at the top.

The differences between Emacs versions 18 and 19 was rather dramatic; the introduction of frames, faces, and colors on windowing systems was obvious to even the most casual user.

There are differences between Emacs versions 19 and 20 as well, but many are more subtle or harder to find. Among the changes are the inclusion of MULE code for languages that use non-Latin characters and for mixing several languages in the same document; the “Customize” facility for modifying variables without having to use Lisp; and automatic conversion of files from Macintosh, Microsoft, and Unix platforms.

A number of older Lisp packages, such as Gnus, Supercite and the calendar/diary, have been updated and enhanced to work with Emacs 20, and are now included with the standard distribution.

4.4 What is different about Emacs 21?

Emacs 21 features a thorough rewrite of the display engine. The new display engine supports variable-size fonts, images, and can play sounds on platforms which support that. As a result, the visual appearence of Emacs, when it runs on a windowed display, is much more reminiscent of modern GUI programs, and includes 3D widgets (used for the mode line and the scroll bars), a configurable and extensible toolbar, tooltips (a.k.a. balloon help), and other niceties.

In addition, Emacs 21 supports faces on text-only terminals. This means that you can now have colors when you run Emacs on a GNU/Linux console and on `xterm` with *`emacs -nw`*.

5 Common requests

5.1 How do I set up a '.emacs' file properly?

See Info file '`emacs`', node '`Init File`'

In general, new Emacs users should not have '`.emacs`' files, because it causes confusing non-standard behavior. Then they send questions to `help-gnu-emacs@gnu.org` asking why Emacs isn't behaving as documented.

Beginning with version 20.1, Emacs includes the new Customize facility, which can be invoked using *M-x customize* ⟨RET⟩. This allows users who are unfamiliar with Emacs Lisp to modify their '`.emacs`' files in a relatively straightforward way, using menus rather than Lisp code. Not all packages support Customize as of this writing, but the number is growing fairly steadily.

While Customize might indeed make it easier to configure Emacs, consider taking a bit of time to learn Emacs Lisp and modifying your '`.emacs`' directly. Simple configuration options are described rather completely in See Info file '`emacs`', node '`Init File`', for users interested in performing frequently requested, basic tasks.

5.2 How do I get colors and syntax highlighting on a TTY?

In Emacs 21.1 and later, colors and faces are supported in non-windowed mode, i.e. on Unix and GNU/Linux text-only terminals and consoles, and when invoked as '`emacs -nw`' on X and MS-Windows. (Colors and faces were supported in the MS-DOS port since Emacs 19.29.) Emacs automatically detects color support at startup and uses it if available. If you think that your terminal supports colors, but Emacs won't use them, check the `termcap` entry for your display type for color-related capabilities.

The command *M-x list-colors-display* pops up a window which exhibits all the colors Emacs knows about on the current display.

Syntax highlighting is usually turned off by default; see Section 5.44 [Turning on syntax highlighting], page 34, for instructions how to turn it on.

5.3 How do I debug a '.emacs' file?

Start Emacs with the '`-debug-init`' command-line option. This enables the Emacs Lisp debugger before evaluating your '`.emacs`' file, and places you in the debugger if something goes wrong. The top line in the '`trace-back`' buffer will be the error message, and the second or third line of that buffer will display the Lisp code from your '`.emacs`' file that caused the problem.

You can also evaluate an individual function or argument to a function in your '`.emacs`' file by moving the cursor to the end of the function or argument and typing *C-x C-e* (*M-x eval-last-sexp*).

Use *C-h v* (*M-x describe-variable*) to check the value of variables which you are trying to set or use.

5.4 How do I make Emacs display the current line (or column) number?

To have Emacs automatically display the current line number of the point in the mode line, do *M-x line-number-mode*. You can also put the form

```
(setq line-number-mode t)
```

in your '.emacs' file to achieve this whenever you start Emacs. (Line number display is on by default, unless your site-specific initialization disables it.) Note that Emacs will not display the line number if the buffer's size in bytes is larger than the value of the variable `line-number-display-limit`.

As of Emacs 20, you can similarly display the current column with *M-x column-number-mode*, or by putting the form

```
(setq column-number-mode t)
```

in your '.emacs' file.

The `"%c"` format specifier in the variable `mode-line-format` will insert the current column's value into the mode line. See the documentation for `mode-line-format` (using *C-h v mode-line-format* ⟨RET⟩) for more information on how to set and use this variable.

Users of all Emacs versions can display the current column using the 'column' package written by Per Abrahamsen (abraham@dina.kvl.dk). See Section 8.3 [Packages that do not come with Emacs], page 50, for instructions on how to get it.

None of the `vi` emulation modes provide the "set number" capability of `vi` (as far as we know).

5.5 How can I modify the titlebar to contain the current file name?

The contents of an Emacs frame's titlebar is controlled by the variable `frame-title-format`, which has the same structure as the variable `mode-line-format`. (Use *C-h v* or *M-x describe-variable* to get information about one or both of these variables.)

By default, the titlebar for a frame does contain the name of the buffer currently being visited, except if there is a single frame. In such a case, the titlebar contains Emacs invocation name and the name of the machine at which Emacs was invoked. This is done by setting `frame-title-format` to the default value of

```
(multiple-frames "%b" ("" invocation-name "@" system-name))
```

To modify the behavior such that frame titlebars contain the buffer's name regardless of the number of existing frames, include the following in your '.emacs':

```
(setq frame-title-format "%b")
```

5.6 How do I turn on abbrevs by default just in mode *mymode*?

Put this in your '.emacs' file:

```
(condition-case ()
   (quietly-read-abbrev-file)
  (file-error nil))
```

```
(add-hook 'mymode-mode-hook
          (lambda ()
           (setq abbrev-mode t)))
```

5.7 How do I turn on `auto-fill-mode` by default?

To turn on `auto-fill-mode` just once for one buffer, use *M-x auto-fill-mode*.

To turn it on for every buffer in a certain mode, you must use the hook for that mode. For example, to turn on `auto-fill` mode for all text buffers, including the following in your '`.emacs`' file:

```
(add-hook 'text-mode-hook 'turn-on-auto-fill)
```

If you want `auto-fill` mode on in all major modes, do this:

```
(setq-default auto-fill-function 'do-auto-fill)
```

5.8 How do I make Emacs use a certain major mode for certain files?

If you want to use a certain mode *foo* for all files whose names end with the extension '.*bar*', this will do it for you:

```
(setq auto-mode-alist (cons '("\\.bar\\'" . foo-mode) auto-mode-alist))
```

Otherwise put this somewhere in the first line of any file you want to edit in the mode *foo* (in the second line, if the first line begins with '`#!`'):

```
-*- foo -*-
```

Beginning with Emacs 19, the variable `interpreter-mode-alist` specifies which mode to use when loading a shell script. (Emacs determines which interpreter you're using by examining the first line of the script.) This feature only applies when the file name doesn't indicate which mode to use. Use *C-h v* (or *M-x describe-variable*) on `interpreter-mode-alist` to learn more.

5.9 How do I search for, delete, or replace unprintable (eight-bit or control) characters?

To search for a single character that appears in the buffer as, for example, '`\237`', you can type *C-s C-q 2 3 7*. (This assumes the value of `search-quote-char` is 17 (i.e., *C-q*).) Searching for **all** unprintable characters is best done with a regular expression (*regexp*) search. The easiest regexp to use for the unprintable chars is the complement of the regexp for the printable chars.

- Regexp for the printable chars: '`[\t\n\r\f -~]`'
- Regexp for the unprintable chars: '`[^\t\n\r\f -~]`'

To type these special characters in an interactive argument to `isearch-forward-regexp` or `re-search-forward`, you need to use *C-q*. ('`\t`', '`\n`', '`\r`', and '`\f`' stand respectively for ⟨TAB⟩, ⟨LFD⟩, ⟨RET⟩, and *C-l*.) So, to search for unprintable characters using `re-search-forward`:

M-x re-search-forward ⟨RET⟩ *[^ ⟨TAB⟩ C-q ⟨LFD⟩ C-q ⟨RET⟩ C-q C-l ⟨SPC⟩ -~]* ⟨RET⟩

Using `isearch-forward-regexp`:

M-C-s [^ ⟨TAB⟩ ⟨LFD⟩ C-q ⟨RET⟩ C-q C-l ⟨SPC⟩ -~]

To delete all unprintable characters, simply use replace-regexp:

M-x replace-regexp ⟨RET⟩ *[^ ⟨TAB⟩ C-q ⟨LFD⟩ C-q ⟨RET⟩ C-q C-l ⟨SPC⟩ -~]* ⟨RET⟩ ⟨RET⟩

Replacing is similar to the above. To replace all unprintable characters with a colon, use:

M-x replace-regexp ⟨RET⟩ [^ ⟨TAB⟩ C-q ⟨LFD⟩ C-q ⟨RET⟩ C-q C-l ⟨SPC⟩ -~] ⟨RET⟩ : ⟨RET⟩

- You don't need to quote ⟨TAB⟩ with either isearch or typing something in the minibuffer.

5.10 How can I highlight a region of text in Emacs?

You can cause the region to be highlighted when the mark is active by including

```
(transient-mark-mode t)
```

in your '`.emacs`' file. (Also see Section 5.44 [Turning on syntax highlighting], page 34.)

5.11 How do I control Emacs's case-sensitivity when searching/replacing?

For searching, the value of the variable `case-fold-search` determines whether they are case sensitive:

```
(setq case-fold-search nil) ; make searches case sensitive
(setq case-fold-search t)   ; make searches case insensitive
```

Similarly, for replacing, the variable `case-replace` determines whether replacements preserve case.

To change the case sensitivity just for one major mode, use the major mode's hook. For example:

```
(add-hook 'foo-mode-hook
          (lambda ()
           (setq case-fold-search nil)))
```

5.12 How do I make Emacs wrap words for me?

Use `auto-fill-mode`, activated by typing *M-x auto-fill-mode*. The default maximum line width is 70, determined by the variable `fill-column`. To learn how to turn this on automatically, see Section 5.7 [Turning on auto-fill by default], page 21.

5.13 Where can I get a better spelling checker for Emacs?

Use Ispell. See Section 9.7 [Ispell], page 57.

5.14 How can I spell-check TeX or *roff documents?

Use Ispell. Ispell can handle TeX and *roff documents. See Section 9.7 [Ispell], page 57.

5.15 How do I change `load-path`?

In general, you should only add to the `load-path`. You can add directory */dir/subdir* to the load path like this:

```
(setq load-path (cons "/dir/subdir/" load-path))
```

To do this relative to your home directory:

```
(setq load-path (cons "~/mysubdir/" load-path)
```

5.16 How do I use an already running Emacs from another window?

`emacsclient`, which comes with Emacs, is for editing a file using an already running Emacs rather than starting up a new Emacs. It does this by sending a request to the already running Emacs, which must be expecting the request.

- Setup:

  Emacs must have executed the `server-start` function for '`emacsclient`' to work. This can be done either by a command line option:

  ```
  emacs -f server-start
  ```

  or by invoking `server-start` from '`.emacs`':

  ```
  (if (some conditions are met) (server-start))
  ```

  When this is done, Emacs starts a subprocess running a program called '`emacsserver`'. '`emacsserver`' creates a Unix domain socket. The socket is either named '`.emacs_server`', in the user's home directory, or '`esrv-`*userid-systemname*', in the '`/tmp`' directory, depending on how '`emacsserver`' was compiled.

  To get your news reader, mail reader, etc., to invoke '`emacsclient`', try setting the environment variable `EDITOR` (or sometimes `VISUAL`) to the value '`emacsclient`'. You may have to specify the full pathname of the '`emacsclient`' program instead. Examples:

  ```
  # csh commands:
  setenv EDITOR emacsclient

  # using full pathname
  setenv EDITOR /usr/local/emacs/etc/emacsclient

  # sh command:
  EDITOR=emacsclient ; export EDITOR
  ```

- Normal use:

  When '`emacsclient`' is run, it connects to the '`.emacs_server`' socket and passes its command line options to '`server`'. When '`server`' receives these requests, it sends this information to the the Emacs process, which at the next opportunity will visit the files specified. (Line numbers can be specified just like with Emacs.) The user will have to switch to the Emacs window by hand. When the user is done editing a file, the user can type *`C-x #`* (or *`M-x server-edit`*) to indicate this. If there is another buffer requested by `emacsclient`, Emacs will switch to it; otherwise `emacsclient` will exit, signaling the calling program to continue.

'emacsclient' and 'server' must be running on machines which share the same filesystem for this to work. The pathnames that 'emacsclient' specifies should be correct for the filesystem that the Emacs process sees. The Emacs process should not be suspended at the time 'emacsclient' is invoked. On Unix and GNU/Linux systems, 'emacsclient' should either be invoked from another X window, or from a shell window inside Emacs itself, or from another interactive session, e.g., by means of a `screen` program.

There is an enhanced version of 'emacsclient'/server called 'gnuserv', written by Andy Norman (ange@hplb.hpl.hp.com) which is available in the Emacs Lisp Archive (see Section 8.3 [Packages that do not come with Emacs], page 50). 'gnuserv' uses Internet domain sockets, so it can work across most network connections. It also supports the execution of arbitrary Emacs Lisp forms and does not require the client program to wait for completion.

The alpha version of an enhanced 'gnuserv' is available at

ftp://ftp.wellfleet.com/netman/psmith/emacs/gnuserv-2.1alpha.tar.gz

5.17 How do I make Emacs recognize my compiler's funny error messages?

The variable `compilation-error-regexp-alist` helps control how Emacs parses your compiler output. It is a list of triplets of the form: (*regexp file-idx line-idx*), where *regexp*, *file-idx* and *line-idx* are strings. To help determine what the constituent elements should be, load 'compile.el' and then type *C-h v compilation-error-regexp-alist* ⟨RET⟩ to see the current value. A good idea is to look at 'compile.el' itself as the comments included for this variable are quite useful—the regular expressions required for your compiler's output may be very close to one already provided. Once you have determined the proper regexps, use the following to inform Emacs of your changes:

```
(setq compilation-error-regexp-alist
      (cons '(regexp file-idx line-idx)
         compilation-error-regexp-alist))
```

5.18 How do I change the indentation for `switch`?

Many people want to indent their `switch` statements like this:

```
f()
{
  switch(x) {
    case A:
      x1;
      break;
    case B:
      x2;
      break;
    default:
      x3;
  }
```

```
}
```

The solution at first appears to be: set `c-indent-level` to 4 and `c-label-offset` to -2. However, this will give you an indentation spacing of four instead of two.

The *real* solution is to use `cc-mode` (the default mode for C programming in Emacs 20 and later) and add the following line to yoyr '`.emacs`':

```
(c-set-offset 'case-label '+)
```

There appears to be no way to do this with the old `c-mode`.

5.19 How to customize indentation in C, C++, and Java buffers?

The Emacs `cc-mode` features an interactive procedure for customizing the indentation style, which is fully explained in the *CC Mode* manual that is part of the Emacs distribution, see section "Customization Indentation" in *The CC Mode Manual*. Here's a short summary of the procedure:

1. Go to the beginning of the first line where you don't like the indentation and type *`C-c C-o`*. Emacs will prompt you for the syntactic symbol; type ⟨RET⟩ to accept the default it suggests.
2. Emacs now prompts for the offset of this syntactic symbol, showing the default (the current definition) inside parentheses. You can choose one of these:

   `0` No extra indentation.

   `+` Indent one basic offset.

   `-` Outdent one basic offset.

   `++` Indent two basic offsets

   `--` Outdent two basic offsets.

   `*` Indent half basic offset.

   `/` Outdent half basic offset.
3. After choosing one of these symbols, type *`C-c C-q`* to reindent the line or the block according to what you just specified.
4. If you don't like the result, go back to step 1. Otherwise, add the following line to your '`.emacs`':

   `(c-set-offset '`*syntactic-symbol offset*`)`

   where *syntactic-symbol* is the name Emacs shows in the minibuffer when you type *`C-c C-o`* at the beginning of the line, and *offset* is one of the indentation symbols listed above (`+`, `/`, `0`, etc.) that you've chosen during the interactive procedure.
5. Go to the next line whose indentation is not to your liking and repeat the process there.

It is recommended to put all the resulting `(c-set-offset ...)` customizations inside a C mode hook, like this:

```
(defun my-c-mode-hook ()
  (c-set-offset ...)
  (c-set-offset ...))
```

```
(add-hook 'c-mode-hook 'my-c-mode-hook)
```

Using `c-mode-hook` avoids the need to put a `(require 'cc-mode)` into your '`.emacs`' file, because `c-set-offset` might be unavailable when `cc-mode` is not loaded.

Note that `c-mode-hook` runs for C source files only; use `c++-mode-hook` for C++ sources, `java-mode-hook` for Java sources, etc. If you want the same customizations to be in effect in *all* languages supported by `cc-mode`, use `c-mode-common-hook`.

5.20 How can I make Emacs automatically scroll horizontally?

In Emacs 21 and later, this is on by default: if the variable `truncate-lines` is non-`nil` in the current buffer, Emacs automatically scrolls the display horizontally when point moves off the left or right edge of the window.

In Emacs 20, use the `hscroll-mode`. Here is some information from the documentation, available by typing *C-h f hscroll-mode* ⟨RET⟩:

Automatically scroll horizontally when the point moves off the left or right edge of the window.

- Type *M-x hscroll-mode* to enable it in the current buffer.
- Type *M-x hscroll-global-mode* to enable it in every buffer.
- `turn-on-hscroll` is useful in mode hooks as in:

  ```
  (add-hook 'text-mode-hook 'turn-on-hscroll)
  ```
- `hscroll-margin` controls how close the cursor can get to the edge of the window.
- `hscroll-step-percent` controls how far to jump once we decide to do so.

5.21 How do I make Emacs "typeover" or "overwrite" instead of inserting?

M-x overwrite-mode (a minor mode). This toggles `overwrite-mode` on and off, so exiting from `overwrite-mode` is as easy as another *M-x overwrite-mode*.

On some systems, ⟨Insert⟩ toggles `overwrite-mode` on and off.

5.22 How do I stop Emacs from beeping on a terminal?

Martin R. Frank (`martin@cc.gatech.edu`) writes:

Tell Emacs to use the *visible bell* instead of the audible bell, and set the visible bell to nothing.

That is, put the following in your `TERMCAP` environment variable (assuming you have one):

```
... :vb=: ...
```

And evaluate the following Lisp form:

```
(setq visible-bell t)
```

5.23 How do I turn down the bell volume in Emacs running under X?

On X Window system, you can adjust the bell volume and duration for all programs with the shell command `xset`.

Invoking `xset` without any arguments produces some basic information, including the following:

```
usage:  xset [-display host:dpy] option ...
  To turn bell off:
      -b                b off               b 0
  To set bell volume, pitch and duration:
       b [vol [pitch [dur]]]          b on
```

5.24 How do I tell Emacs to automatically indent a new line to the indentation of the previous line?

Such behavior is automatic in Emacs 20 and later. From the 'etc/NEWS' file for Emacs 20.2:

```
** In Text mode, now only blank lines separate paragraphs.  This makes
it possible to get the full benefit of Adaptive Fill mode in Text mode,
and other modes derived from it (such as Mail mode).  <TAB> in Text
mode now runs the command indent-relative; this makes a practical
difference only when you use indented paragraphs.

As a result, the old Indented Text mode is now identical to Text mode,
and is an alias for it.

If you want spaces at the beginning of a line to start a paragraph, use
the new mode, Paragraph Indent Text mode.
```

If you have `auto-fill-mode` turned on (see Section 5.7 [Turning on auto-fill by default], page 21), you can tell Emacs to prefix every line with a certain character sequence, the *fill prefix*. Type the prefix at the beginning of a line, position point after it, and then type *C-x .* (`set-fill-prefix`) to set the fill prefix. Thereafter, auto-filling will automatically put the fill prefix at the beginning of new lines, and *M-q* (`fill-paragraph`) will maintain any fill prefix when refilling the paragraph.

If you have paragraphs with different levels of indentation, you will have to set the fill prefix to the correct value each time you move to a new paragraph. To avoid this hassle, try one of the many packages available from the Emacs Lisp Archive (see Section 8.3 [Packages that do not come with Emacs], page 50.) Look up "fill" and "indent" in the Lisp Code Directory for guidance.

5.25 How do I show which parenthesis matches the one I'm looking at?

As of version 19, Emacs comes with 'paren.el', which (when loaded) will automatically highlight matching parentheses whenever point (i.e., the cursor) is located over one. To load 'paren.el' automatically, include the line

```
(require 'paren)
```

in your '.emacs' file. Alan Shutko (shutkoa@ugsolutions.com) reports that as of version 20.1, you must also call show-paren-mode in your '.emacs' file:

```
(show-paren-mode 1)
```

Customize will let you turn on show-paren-mode. Use *M-x customize-group* ⟨RET⟩ *paren-showing* ⟨RET⟩. From within Customize, you can also go directly to the "paren-showing" group.

Alternatives to paren include:

- If you're looking at a right parenthesis (or brace or bracket) you can delete it and reinsert it. Emacs will momentarily move the cursor to the matching parenthesis.
- *M-C-f* (forward-sexp) and *M-C-b* (backward-sexp) will skip over one set of balanced parentheses, so you can see which parentheses match. (You can train it to skip over balanced brackets and braces at the same time by modifying the syntax table.)
- Here is some Emacs Lisp that will make the ⟨%⟩ key show the matching parenthesis, like in vi. In addition, if the cursor isn't over a parenthesis, it simply inserts a % like normal.

```
;; By an unknown contributor

(global-set-key "%" 'match-paren)

(defun match-paren (arg)
  "Go to the matching paren if on a paren; otherwise insert %."
  (interactive "p")
  (cond ((looking-at "\\s\(") (forward-list 1) (backward-char 1))
        ((looking-at "\\s\)") (forward-char 1) (backward-list 1))
        (t (self-insert-command (or arg 1)))))
```

5.26 In C mode, can I show just the lines that will be left after #ifdef commands are handled by the compiler?

M-x hide-ifdef-mode. (This is a minor mode.) You might also want to try 'cpp.el', available at the Emacs Lisp Archive (see Section 8.3 [Packages that do not come with Emacs], page 50).

5.27 Is there an equivalent to the . (dot) command of vi?

(. is the redo command in vi. It redoes the last insertion/deletion.)

As of Emacs 20.3, there is indeed a repeat command (*C-x z*) that repeats the last command. If you preface it with a prefix argument, the prefix arg is applied to the command.

You can also type *C-x* ⟨ESC⟩ ⟨ESC⟩ (repeat-complex-command) to reinvoke commands that used the minibuffer to get arguments. In repeat-complex-command you can type *M-p* and *M-n* (and also up-arrow and down-arrow, if your keyboard has these keys) to scan through all the different complex commands you've typed.

To repeat a set of commands, use keyboard macros. (See Info file 'emacs', node 'Keyboard Macros'.)

If you're really desperate for the `.` command, use VIPER, a `vi` emulation mode which comes with Emacs, and which appears to support it. (See Section 9.4 [VIPER], page 56.)

5.28 What are the valid X resource settings (i.e., stuff in .Xdefaults)?

See Info file '`emacs`', node '`Resources X`'.

You can also use a resource editor, such as editres (for X11R5 and onwards), to look at the resource names for the menu bar, assuming Emacs was compiled with the X toolkit.

5.29 How do I execute ("evaluate") a piece of Emacs Lisp code?

There are a number of ways to execute (*evaluate*, in Lisp lingo) an Emacs Lisp *form*:

- If you want it evaluated every time you run Emacs, put it in a file named '`.emacs`' in your home directory. This is known as "your '`.emacs`' file," and contains all of your personal customizations.
- You can type the form in the '`*scratch*`' buffer, and then type ⟨LFD⟩ (or *C-j*) after it. The result of evaluating the form will be inserted in the buffer.
- In `emacs-lisp-mode`, typing *M-C-x* evaluates a top-level form before or around point.
- Typing *C-x C-e* in any buffer evaluates the Lisp form immediately before point and prints its value in the echo area.
- Typing *M-:* or *M-x eval-expression* allows you to type a Lisp form in the minibuffer which will be evaluated once you press ⟨RET⟩.
- You can use *M-x load-file* to have Emacs evaluate all the Lisp forms in a file. (To do this from Lisp use the function `load` instead.)

  The functions `load-library`, `eval-region`, `eval-current-buffer`, `require`, and `autoload` are also useful; see Section 3.4 [Emacs Lisp documentation], page 12, if you want to learn more about them.

5.30 How do I change Emacs's idea of the ⟨TAB⟩ character's length?

Set the variable `default-tab-width`. For example, to set ⟨TAB⟩ stops every 10 characters, insert the following in your '`.emacs`' file:

```
(setq default-tab-width 10)
```

Do not confuse variable `tab-width` with variable `tab-stop-list`. The former is used for the display of literal ⟨TAB⟩ characters. The latter controls what characters are inserted when you press the ⟨TAB⟩ character in certain modes.

5.31 How do I insert '>' at the beginning of every line?

To do this to an entire buffer, type *M-< M-x replace-regexp* ⟨RET⟩ `^` ⟨RET⟩ `>` ⟨RET⟩.

To do this to a region, use `string-insert-rectangle`. Set the mark (*C-*⟨SPC⟩) at the beginning of the first line you want to prefix, move the cursor to last line to be prefixed, and type *M-x string-insert-rectangle* ⟨RET⟩. To do this for the whole buffer, type *C-x h M-x string-insert-rectangle* ⟨RET⟩.

If you are trying to prefix a yanked mail message with '>', you might want to set the variable `mail-yank-prefix`. Better yet, use the Supercite package (see Section 9.2 [Supercite], page 55), which provides flexible citation for yanked mail and news messages; it is included in Emacs since version 19.20. See Section 12.1 [Changing the included text prefix], page 69, for additional information.

5.32 How do I insert "_^H" before each character in a region to get an underlined paragraph?

Mark the region and then type *M-x underline-region* ⟨RET⟩.

5.33 How do I repeat a command as many times as possible?

Use *C-x (* and *C-x)* to make a keyboard macro that invokes the command and then type *M-0 C-x e*.

Any messages your command prints in the echo area will be suppressed.

If you need to repeat a command a small number of times, you can use *C-x z*, see Section 5.27 [Repeating commands], page 28.

5.34 How do I make Emacs behave like this: when I go up or down, the cursor should stay in the same column even if the line is too short?

M-x picture-mode.

5.35 How do I tell Emacs to iconify itself?

C-z iconifies Emacs when running under X and suspends Emacs otherwise. See Info file '`emacs`', node '`Misc X`'.

5.36 How do I use regexps (regular expressions) in Emacs?

See Info file '`emacs`', node '`Regexps`'.

The `or` operator is '`\|`', not '`|`', and the grouping operators are '`\(`' and '`\)`'. Also, the string syntax for a backslash is '`\\`'. To specify a regular expression like '`xxx\(foo\|bar\)`' in a Lisp string, use '`xxx\\(foo\\|bar\\)`'.

Note the doubled backslashes!

- Unlike in Unix '`grep`', '`sed`', etc., a complement character set ('`[^...]`') can match a newline character (⟨LFD⟩ a.k.a. *C-j* a.k.a. '`\n`'), unless newline is mentioned as one of the characters not to match.
- The character syntax regexps (e.g., '`\sw`') are not meaningful inside character set regexps (e.g., '`[aeiou]`'). (This is actually typical for regexp syntax.)

5.37 How do I perform a replace operation across more than one file?

The "tags" feature of Emacs includes the command `tags-query-replace` which performs a query-replace across all the files mentioned in the '`TAGS`' file. See Info file '`emacs`', node '`Tags Search`'.

As of Emacs 19.29, Dired mode (*M-x dired* ⟨RET⟩, or *C-x d*) supports the command `dired-do-query-replace`, which allows users to replace regular expressions in multiple files.

5.38 Where is the documentation for `etags`?

The `etags` man page should be in the same place as the `emacs` man page.

Quick command-line switch descriptions are also available. For example, '`etags -H`'.

5.39 How do I disable backup files?

You probably don't want to do this, since backups are useful, especially when something goes wrong.

To avoid seeing backup files (and other "uninteresting" files) in Dired, load `dired-x` by adding the following to your '`.emacs`' file:

```
(add-hook 'dired-load-hook
          (lambda ()
           (load "dired-x")))
```

With `dired-x` loaded, *M-o* toggles omitting in each dired buffer. You can make omitting the default for new dired buffers by putting the following in your '`.emacs`':

```
(add-hook 'dired-mode-hook 'dired-omit-toggle)
```

If you're tired of seeing backup files whenever you do an '`ls`' at the Unix shell, try GNU `ls` with the '`-B`' option. GNU `ls` is part of the GNU Fileutils package, available from '`ftp.gnu.org`' and its mirrors (see Section 8.5 [Current GNU distributions], page 50).

To disable or change the way backups are made, See Info file '`emacs`', node '`Backup Names`'.

Beginning with Emacs 21.1, you can control where Emacs puts backup files by customizing the variable `backup-directory-alist`. This variable's value specifies that files whose names match specific patters should have their backups put in certain directories. A typical use is to add the element `("." . dir)` to force Emacs to put **all** backup files in the directory '`dir`'.

5.40 How do I disable `auto-save-mode`?

You probably don't want to do this, since auto-saving is useful, especially when Emacs or your computer crashes while you are editing a document.

Instead, you might want to change the variable `auto-save-interval`, which specifies how many keystrokes Emacs waits before auto-saving. Increasing this value forces Emacs to wait longer between auto-saves, which might annoy you less.

You might also want to look into Sebastian Kremer's `auto-save` package, available from the Lisp Code Archive (see Section 8.3 [Packages that do not come with Emacs], page 50). This package also allows you to place all auto-save files in one directory, such as '`/tmp`'.

To disable or change how `auto-save-mode` works, See Info file '`emacs`', node '`Auto Save`'.

5.41 How can I go to a certain line given its number?

Are you sure you indeed need to go to a line by its number? Perhaps all you want is to display a line in your source file for which a compiler printed an error message? If so, compiling from within Emacs using the *M-x compile* and *M-x recompile* commands is a much more effective way of doing that. Emacs automatically intercepts the compile error messages, inserts them into a special buffer called `*compilation*`, and lets you visit the locus of each message in the source. Type *C-x `* to step through the offending lines one by one. Click *Mouse-2* or press ⟨RET⟩ on a message text in the `*compilation*` buffer to go to the line whose number is mentioned in that message.

But if you indeed need to go to a certain text line, type *M-x goto-line* ⟨RET⟩. Emacs will prompt you for the number of the line and go to that line.

You can do this faster by invoking `goto-line` with a numeric argument that is the line's number. For example, *C-u 286 M-x goto-line* ⟨RET⟩ will jump to line number 286 in the current buffer.

If you need to use this command frequently, you might consider binding it to a key. The following snippet, if added to your '`~/.emacs`' file, will bind the sequence *C-x g* to `goto-line`:

```
(global-set-key "\C-xg" 'goto-line)
```

5.42 How can I create or modify new pull-down menu options?

Each menu title (e.g., '`File`', '`Edit`', '`Buffers`') represents a local or global keymap. Selecting a menu title with the mouse displays that keymap's non-nil contents in the form of a menu.

So to add a menu option to an existing menu, all you have to do is add a new definition to the appropriate keymap. Adding a '`Forward Word`' item to the '`Edit`' menu thus requires the following Lisp code:

```
(define-key global-map
  [menu-bar edit forward]
  '("Forward word" . forward-word))
```

The first line adds the entry to the global keymap, which includes global menu bar entries. Replacing the reference to `global-map` with a local keymap would add this menu option only within a particular mode.

The second line describes the path from the menu-bar to the new entry. Placing this menu entry underneath the '`File`' menu would mean changing the word `edit` in the second line to `file`.

The third line is a cons cell whose first element is the title that will be displayed, and whose second element is the function that will be called when that menu option is invoked.

To add a new menu, rather than a new option to an existing menu, we must define an entirely new keymap:

```
(define-key global-map [menu-bar words]
  (cons "Words" (make-sparse-keymap "Words")))
```

The above code creates a new sparse keymap, gives it the name '`Words`', and attaches it to the global menu bar. Adding the '`Forward Word`' item to this new menu would thus require the following code:

```
(define-key global-map
  [menu-bar words forward]
  '("Forward word" . forward-word))
```

Note that because of the way keymaps work, menu options are displayed with the more recently defined items at the top. Thus if you were to define menu options '`foo`', '`bar`', and '`baz`' (in that order), the menu option '`baz`' would appear at the top, and '`foo`' would be at the bottom.

One way to avoid this problem is to use the function `define-key-after`, which works the same as `define-key`, but lets you modify where items appear. The following Lisp code would insert the '`Forward Word`' item in the '`Edit`' menu immediately following the '`Undo`' item:

```
(define-key-after
  (lookup-key global-map [menu-bar edit])
  [forward]
  '("Forward word" . forward-word)
  'undo)
```

Note how the second and third arguments to `define-key-after` are different from those of `define-key`, and that we have added a new (final) argument, the function after which our new key should be defined.

To move a menu option from one position to another, simply evaluate `define-key-after` with the appropriate final argument.

More detailed information—and more examples of how to create and modify menu options—are in the *Emacs Lisp Reference Manual*, under "Menu Keymaps". (See Section 3.4 [Emacs Lisp documentation], page 12, for information on this manual.)

5.43 How do I delete menus and menu options?

The simplest way to remove a menu is to set its keymap to '`nil`'. For example, to delete the '`Words`' menu (see Section 5.42 [Modifying pull-down menus], page 32), use:

```
(define-key global-map [menu-bar words] nil)
```

Similarly, removing a menu option requires redefining a keymap entry to `nil`. For example, to delete the '`Forward word`' menu option from the '`Edit`' menu (we added it in Section 5.42 [Modifying pull-down menus], page 32), use:

```
(define-key global-map [menu-bar edit forward] nil)
```

5.44 How do I turn on syntax highlighting?

`font-lock-mode` is the standard way to have Emacs perform syntax highlighting in the current buffer. With `font-lock-mode` turned on, different types of text will appear in different colors. For instance, if you turn on `font-lock-mode` in a programming mode, variables will appear in one face, keywords in a second, and comments in a third.

Earlier versions of Emacs supported hilit19, a similar package. Use of hilit19 is now considered non-standard, although '`hilit19.el`' comes with the stock Emacs distribution. It is no longer maintained.

To turn `font-lock-mode` on within an existing buffer, use *`M-x font-lock-mode`* ⟨RET⟩.

To automatically invoke `font-lock-mode` when a particular major mode is invoked, set the major mode's hook. For example, to fontify all `c-mode` buffers, add the following to your '`.emacs`' file:

```
(add-hook 'c-mode-hook 'turn-on-font-lock)
```

To automatically invoke `font-lock-mode` for all major modes, you can turn on `global-font-lock-mode` by including the following line in your '`.emacs`' file:

```
(global-font-lock-mode 1)
```

This instructs Emacs to turn on font-lock mode in those buffers for which a font-lock mode definition has been provided (in the variable `font-lock-global-modes`). If you edit a file in `pie-ala-mode`, and no font-lock definitions have been provided for `pie-ala` files, then the above setting will have no effect on that particular buffer.

Highlighting a buffer with `font-lock-mode` can take quite a while, and cause an annoying delay in display, so several features exist to work around this.

In Emacs 21 and later, turning on `font-lock-mode` automatically activates the new *Just-In-Time fontification* provided by `jit-lock-mode`. `jit-lock-mode` defers the fontification of portions of buffer until you actually need to see them, and can also fontify while Emacs is idle. This makes display of the visible portion of a buffer almost instantaneous. For details about customizing `jit-lock-mode`, type *`C-h f jit-lock-mode`* ⟨RET⟩.

In versions of Emacs before 21, different levels of decoration are available, from slight to gaudy. More decoration means you need to wait more time for a buffer to be fontified (or a faster machine). To control how decorated your buffers should become, set the value of `font-lock-maximum-decoration` in your '`.emacs`' file, with a `nil` value indicating default (usually minimum) decoration, and a `t` value indicating the maximum decoration. For the gaudiest possible look, then, include the line

```
(setq font-lock-maximum-decoration t)
```

in your '`.emacs`' file. You can also set this variable such that different modes are highlighted in a different ways; for more information, see the documentation for `font-lock-maximum-decoration` with *`C-h v`* (or *`M-x describe-variable`* ⟨RET⟩).

You might also want to investigate `fast-lock-mode` and `lazy-lock-mode`, versions of `font-lock-mode` that speed up highlighting. These are the alternatives for `jit-lock-mode` in versions of Emacs before 21.1. The advantage of `lazy-lock-mode` is that it only fontifies buffers when certain conditions are met, such as after a certain amount of idle time, or after you have finished scrolling through text. See the documentation for `lazy-lock-mode` by typing *C-h f* `lazy-lock-mode` (*M-x describe-function* ⟨RET⟩ *lazy-lock-mode* ⟨RET⟩).

Also see the documentation for the function `font-lock-mode`, available by typing *C-h f font-lock-mode* (*M-x describe-function* ⟨RET⟩ *font-lock-mode* ⟨RET⟩).

For more information on font-lock mode, take a look at the `font-lock-mode` FAQ, maintained by Jari Aalto (`jari.aalto@ntc.nokia.com`) at

`ftp://cs.uta.fi/pub/ssjaaa/ema-font.gui`

To print buffers with the faces (i.e., colors and fonts) intact, use *M-x ps-print-buffer-with-faces* or *M-x ps-print-region-with-faces*. You will need a way to send text to a PostScript printer, or a PostScript interpreter such as Ghostscript; consult the documentation of the variables `ps-printer-name`, `ps-lpr-command`, and `ps-lpr-switches` for more details.

5.45 How can I force Emacs to scroll only one line when I move past the bottom of the screen?

Place the following Lisp form in your '`.emacs`' file:

```
(setq scroll-step 1)
```

See Info file '`emacs`', node '`Scrolling`'.

5.46 How can I replace highlighted text with what I type?

Use `delete-selection-mode`, which you can start automatically by placing the following Lisp form in your '`.emacs`' file:

```
(delete-selection-mode t)
```

According to the documentation string for `delete-selection-mode` (which you can read using *M-x describe-function* ⟨RET⟩ *delete-selection-mode* ⟨RET⟩):

> When ON, typed text replaces the selection if the selection is active. When OFF, typed text is just inserted at point.

This mode also allows you to delete (not kill) the highlighted region by pressing ⟨DEL⟩.

5.47 How can I edit MS-DOS files using Emacs?

As of Emacs 20, detection and handling of MS-DOS (and Windows) files is performed transparently. You can open MS-DOS files on a Unix system, edit it, and save it without having to worry about the file format.

When editing an MS-DOS style file, the mode line will indicate that it is a DOS file. On Unix and GNU/Linux systems, and also on a Macintosh, the string '`(DOS)`' will appear near the left edge of the mode line; on DOS and Windows, where the DOS end-of-line (EOL) format is the default, a backslash ('\') will appear in the mode line.

If you are running a version of Emacs before 20.1, get `crypt++` from the Emacs Lisp Archive (see Section 8.3 [Packages that do not come with Emacs], page 50). Among other things, `crypt++` transparently modifies MS-DOS files as they are loaded and saved, allowing you to ignore the different conventions that Unix and MS-DOS have for delineating the end of a line.

5.48 How can I tell Emacs to fill paragraphs with a single space after each period?

Ulrich Mueller (`ulm@vsnhd1.cern.ch`) suggests adding the following two lines to your '`.emacs`' file:

```
(setq sentence-end "[.?!][]\"')}]*\\($\\|[ \t]\\)[ \t\n]*")
(setq sentence-end-double-space nil)
```

5.49 Why do I get these strange escape sequences when I run

`ls` from the Shell mode?

This happens because `ls` is aliased to '`ls --color`' in your shell init file. You have two alternatives to solve this:

- Make the alias conditioned on the `EMACS` variable in the environment. When Emacs runs a subsidiary shell, it exports the `EMACS` variable with the value `t` to that shell. You can unalias `ls` when that happens, thus limiting the alias to your interactive sessions.
- Install the `ansi-color` package (bundled with Emacs 21.1 and later), which converts these ANSI escape sequences into colors.

6 Bugs and problems

The Emacs manual lists some common kinds of trouble users could get into, see section "Dealing with Emacs Trouble" in *The GNU Emacs Manual*, so you might look there if the problem you encounter isn't described in this chapter. If you decide you've discovered a bug, see section "Reporting Bugs" in *The GNU Emacs Manual*, for instructions how to do that.

The file '`etc/PROBLEMS`' in the Emacs distribution lists various known problems with building and using Emacs on specific platforms; type *`C-h P`* to read it.

6.1 Does Emacs have problems with files larger than 8 megabytes?

Old versions (i.e., anything before 19.29) of Emacs had problems editing files larger than 8 megabytes. As of version 19.29, the maximum buffer size is at least 2^27-1, or 134,217,727 bytes, or 132 MBytes. Emacs 20 can be compiled on some 64-bit systems in a way that enlarges the buffer size up to 576,460,752,303,423,487 bytes, or 549,755,813 GBytes.

If you are using a version of Emacs older than 19.29 and cannot upgrade, you will have to recompile. Leonard N. Zubkoff (`lnz@lucid.com`) suggests putting the following two lines in '`src/config.h`' before compiling Emacs to allow for 26-bit integers and pointers (and thus file sizes of up to 33,554,431 bytes):

```
#define VALBITS 26
#define GCTYPEBITS 5
```

This method may result in "ILLEGAL DATATYPE" and other random errors on some machines.

David Gillespie (`daveg@csvax.cs.caltech.edu`) explains how this problems crops up; while his numbers are true only for pre-19.29 versions of Emacs, the theory remains the same with current versions.

> Emacs is largely written in a dialect of Lisp; Lisp is a freely-typed language in the sense that you can put any value of any type into any variable, or return it from a function, and so on. So each value must carry a *tag* along with it identifying what kind of thing it is, e.g., integer, pointer to a list, pointer to an editing buffer, and so on. Emacs uses standard 32-bit integers for data objects, taking the top 8 bits for the tag and the bottom 24 bits for the value. So integers (and pointers) are somewhat restricted compared to true C integers and pointers.

6.2 How do I get rid of '`^M`' or echoed commands in my shell buffer?

Try typing *`M-x shell-strip-ctrl-m`* ⟨RET⟩ while in `shell-mode` to make them go away. If that doesn't work, you have several options:

For `tcsh`, put this in your '`.cshrc`' (or '`.tcshrc`') file:

```
if ($?EMACS) then
    if ("$EMACS" == t) then
```

```
          if ($?tcsh) unset edit
          stty nl
      endif
  endif
```

Or put this in your '.emacs_tcsh' file:

```
unset edit
stty nl
```

Alternatively, use `csh` in your shell buffers instead of `tcsh`. One way is:

```
(setq explicit-shell-file-name "/bin/csh")
```

and another is to do this in your '.cshrc' (or '.tcshrc') file:

```
setenv ESHELL /bin/csh
```

(You must start Emacs over again with the environment variable properly set for this to take effect.)

You can also set the `ESHELL` environment variable in Emacs Lisp with the following Lisp form,

```
(setenv "ESHELL" "/bin/csh")
```

The above solutions try to prevent the shell from producing the '^M' characters in the first place. If this is not possible (e.g., if you use a Windows shell), you can get Emacs to remove these characters from the buffer by adding this to your '.emacs' init file:

```
(add-hook 'comint-output-filter-functions 'shell-strip-ctrl-m)
```

On a related note: If your shell is echoing your input line in the shell buffer, you might want to try the following command in your shell start-up file:

```
stty -icrnl -onlcr -echo susp ^Z
```

6.3 Why do I get "Process shell exited abnormally with code 1"?

The most likely reason for this message is that the 'env' program is not properly installed. Compile this program for your architecture, and install it with 'a+x' permission in the architecture-dependent Emacs program directory. (You can find what this directory is at your site by inspecting the value of the variable `exec-directory` by typing *C-h v exec-directory* ⟨RET⟩.)

You should also check for other programs named 'env' in your path (e.g., SunOS has a program named '/usr/bin/env'). We don't understand why this can cause a failure and don't know a general solution for working around the problem in this case.

The 'make clean' command will remove 'env' and other vital programs, so be careful when using it.

It has been reported that this sometimes happened when Emacs was started as an X client from an xterm window (i.e., had a controlling tty) but the xterm was later terminated.

See also 'PROBLEMS' (in the 'etc' subdirectory of the top-level directory when you unpack the Emacs source) for other possible causes of this message.

6.4 Why do I get an error message when I try to run *M-x shell*?

On MS-Windows, this might happen because Emacs tries to look for the shell in a wrong place. The default file name '`/bin/sh`' is usually incorrect for non-Unix systems. If you know where your shell executable is, set the variable `explicit-shell-file-name` in your '`.emacs`' file to point to its full file name, like this:

```
(setq explicit-shell-file-name "d:/shells/bash.exe")
```

If you don't know what shell does Emacs use, try the *M-!* command; if that works, put the following line into your '`.emacs`':

```
(setq explicit-shell-file-name shell-file-name)
```

Some people have trouble with Shell Mode because of intrusive antivirus software; disabling the resident antivirus program solves the problems in those cases.

6.5 Where is the termcap/terminfo entry for terminal type "emacs"?

The termcap entry for terminal type '`emacs`' is ordinarily put in the '`TERMCAP`' environment variable of subshells. It may help in certain situations (e.g., using rlogin from shell buffer) to add an entry for '`emacs`' to the system-wide termcap file. Here is a correct termcap entry for '`emacs`':

```
emacs:tc=unknown:
```

To make a terminfo entry for '`emacs`', use `tic` or `captoinfo`. You need to generate '`/usr/lib/terminfo/e/emacs`'. It may work to simply copy '`/usr/lib/terminfo/d/dumb`' to '`/usr/lib/terminfo/e/emacs`'.

Having a termcap/terminfo entry will not enable the use of full screen programs in shell buffers. Use *M-x terminal-emulator* for that instead.

A workaround to the problem of missing termcap/terminfo entries is to change terminal type '`emacs`' to type '`dumb`' or '`unknown`' in your shell start up file. `csh` users could put this in their '`.cshrc`' files:

```
if ("$term" == emacs) set term=dumb
```

6.6 Why does Emacs spontaneously start displaying "I-search:" and beeping?

Your terminal (or something between your terminal and the computer) is sending *C-s* and *C-q* for flow control, and Emacs is receiving these characters and interpreting them as commands. (The *C-s* character normally invokes the `isearch-forward` command.) For possible solutions, see Section 10.7 [Handling C-s and C-q with flow control], page 60.

6.7 Why can't Emacs talk to certain hosts (or certain hostnames)?

The problem may be that Emacs is linked with a wimpier version of `gethostbyname` than the rest of the programs on the machine. This is often manifested as a message on startup of "X server not responding. Check your 'DISPLAY' environment variable." or a message of "Unknown host" from `open-network-stream`.

On a Sun, this may be because Emacs had to be linked with the static C library. The version of `gethostbyname` in the static C library may only look in '/etc/hosts' and the NIS (YP) maps, while the version in the dynamic C library may be smart enough to check DNS in addition to or instead of NIS. On a Motorola Delta running System V R3.6, the version of `gethostbyname` in the standard library works, but the one that works with NIS doesn't (the one you get with -linet). Other operating systems have similar problems.

Try these options:

- Explicitly add the host you want to communicate with to '/etc/hosts'.
- Relink Emacs with this line in 'src/config.h':

  ```
  #define LIBS_SYSTEM -lresolv
  ```

- Replace `gethostbyname` and friends in 'libc.a' with more useful versions such as the ones in 'libresolv.a'. Then relink Emacs.
- If you are actually running NIS, make sure that `ypbind` is properly told to do DNS lookups with the correct command line switch.

6.8 Why does Emacs say "Error in init file"?

An error occurred while loading either your '.emacs' file or the system-wide file 'lisp/default.el'. Emacs 21.1 and later pops the '*Messages*' buffer, and puts there some additional information about the error, to provide some hints for debugging.

For information on how to debug your '.emacs' file, see Section 5.3 [Debugging a customization file], page 19.

It may be the case that you need to load some package first, or use a hook that will be evaluated after the package is loaded. A common case of this is explained in Section 10.3 [Terminal setup code works after Emacs has begun], page 60.

6.9 Why does Emacs ignore my X resources (my .Xdefaults file)?

As of version 19, Emacs searches for X resources in the files specified by the following environment variables:

- `XFILESEARCHPATH`
- `XUSERFILESEARCHPATH`
- `XAPPLRESDIR`

This emulates the functionality provided by programs written using the Xt toolkit.

`XFILESEARCHPATH` and `XUSERFILESEARCHPATH` should be a list of file names separated by colons. `XAPPLRESDIR` should be a list of directory names separated by colons.

Emacs searches for X resources:

1. specified on the command line, with the '-xrm RESOURCESTRING' option,
2. then in the value of the 'XENVIRONMENT' environment variable,

   - or if that is unset, in the file named '`~/.Xdefaults-`*hostname*' if it exists (where *hostname* is the name of the machine Emacs is running on),
3. then in the screen-specific and server-wide resource properties provided by the server,
   - or if those properties are unset, in the file named '`~/.Xdefaults`' if it exists,
4. then in the files listed in '`XUSERFILESEARCHPATH`',
   - or in files named '*lang*`/Emacs`' in directories listed in '`XAPPLRESDIR`' (where *lang* is the value of the `LANG` environment variable), if the '`LANG`' environment variable is set,
   - or in files named Emacs in the directories listed in '`XAPPLRESDIR`'
   - or in '`~/`*lang*`/Emacs`' (if the `LANG` environment variable is set),
   - or in '`~/Emacs`',
5. then in the files listed in `XFILESEARCHPATH`.

6.10 Why don't my customizations of the frame parameters work?

This probably happens because you have set the frame parameters in the variable `initial-frame-alist`. That variable holds parameters used only for the first frame created when Emacs starts. To customize the parameters of all frames, change the variable `default-frame-alist` instead.

These two variables exist because many users customize the initial frame in a special way. For example, you could determine the position and size of the initial frame, but would like to control the geometry of the other frames by individually positioning each one of them.

6.11 Why does Emacs take 20 seconds to visit a file?

Old versions of Emacs (i.e., versions before Emacs 20.x) often encountered this when the master lock file, '`!!!SuperLock!!!`', has been left in the lock directory somehow. Delete it.

Mark Meuer (`meuer@geom.umn.edu`) says that NeXT NFS has a bug where an exclusive create succeeds but returns an error status. This can cause the same problem. Since Emacs's file locking doesn't work over NFS anyway, the best solution is to recompile Emacs with `CLASH_DETECTION` undefined.

6.12 How do I edit a file with a '`$`' in its name?

When entering a file name in the minibuffer, Emacs will attempt to expand a '`$`' followed by a word as an environment variable. To suppress this behavior, type *$$* instead.

6.13 Why does shell mode lose track of the shell's current directory?

Emacs has no way of knowing when the shell actually changes its directory. This is an intrinsic limitation of Unix. So it tries to guess by recognizing '`cd`' commands. If you type `cd` followed by a directory name with a variable reference (`cd $HOME/bin`) or with a shell metacharacter (`cd ../lib*`), Emacs will fail to correctly guess the shell's new current directory. A huge variety of fixes and enhancements to shell mode for this problem have been written to handle this problem. Check the Lisp Code Directory (see Section 8.2 [Finding a package with particular functionality], page 49).

You can tell Emacs the shell's current directory with the command `M-x dirs`.

6.14 Are there any security risks in Emacs?

- The '`movemail`' incident. (No, this is not a risk.)

  In his book *The Cuckoo's Egg*, Cliff Stoll describes this in chapter 4. The site at LBL had installed the '`/etc/movemail`' program setuid root. (As of version 19, '`movemail`' is in your architecture-specific directory; type `C-h v exec-directory` ⟨RET⟩ to see what it is.) Since `movemail` had not been designed for this situation, a security hole was created and users could get root privileges.

  `movemail` has since been changed so that this security hole will not exist, even if it is installed setuid root. However, `movemail` no longer needs to be installed setuid root, which should eliminate this particular risk.

  We have heard unverified reports that the 1988 Internet worm took advantage of this configuration problem.

- The `file-local-variable` feature. (Yes, a risk, but easy to change.)

  There is an Emacs feature that allows the setting of local values for variables when editing a file by including specially formatted text near the end of the file. This feature also includes the ability to have arbitrary Emacs Lisp code evaluated when the file is visited. Obviously, there is a potential for Trojan horses to exploit this feature.

  Emacs 18 allowed this feature by default; users could disable it by setting the variable `inhibit-local-variables` to a non-nil value.

  As of Emacs 19, Emacs has a list of local variables that create a security risk. If a file tries to set one of them, it asks the user to confirm whether the variables should be set. You can also tell Emacs whether to allow the evaluation of Emacs Lisp code found at the bottom of files by setting the variable `enable-local-eval`.

  For more information, See Info file '`emacs`', node '`File Variables`'.

- Synthetic X events. (Yes, a risk; use '`MIT-MAGIC-COOKIE-1`' or better.)

  Emacs accepts synthetic X events generated by the `SendEvent` request as though they were regular events. As a result, if you are using the trivial host-based authentication, other users who can open X connections to your X workstation can make your Emacs process do anything, including run other processes with your privileges.

  The only fix for this is to prevent other users from being able to open X connections. The standard way to prevent this is to use a real authentication mechanism, such as '`MIT-MAGIC-COOKIE-1`'. If using the `xauth` program has any effect, then you are probably using '`MIT-MAGIC-COOKIE-1`'. Your site may be using a superior authentication method; ask your system administrator.

  If real authentication is not a possibility, you may be satisfied by just allowing hosts access for brief intervals while you start your X programs, then removing the access.

This reduces the risk somewhat by narrowing the time window when hostile users would have access, but *does not eliminate the risk*.

On most computers running Unix and X, you enable and disable access using the `xhost` command. To allow all hosts access to your X server, use

```
xhost +
```

at the shell prompt, which (on an HP machine, at least) produces the following message:

```
access control disabled, clients can connect from any host
```

To deny all hosts access to your X server (except those explicitly allowed by name), use

```
xhost -
```

On the test HP computer, this command generated the following message:

```
access control enabled, only authorized clients can connect
```

6.15 Dired says, "no file on this line" when I try to do something.

Chances are you're using a localized version of Unix that doesn't use US date format in dired listings. You can check this by looking at dired listings or by typing *`ls -l`* to a shell and looking at the dates that come out.

Dired uses a regular expression to find the beginning of a file name. In a long Unix-style directory listing ('`ls -l`'), the file name starts after the date. The regexp has thus been written to look for the date, the format of which can vary on non-US systems.

There are two approaches to solving this. The first one involves setting things up so that '`ls -l`' outputs US date format. This can be done by setting the locale. See your OS manual for more information.

The second approach involves changing the regular expression used by dired, `dired-move-to-filename-regexp`.

7 Compiling and installing Emacs

7.1 How do I install Emacs?

This answer is meant for users of Unix and Unix-like systems. Users of other operating systems should see the series of questions beginning with Section 8.7 [Emacs for MS-DOS], page 51, which describe where to get non-Unix source and binaries, and how to install Emacs on those systems.

For Unix and Unix-like systems, the easiest way is often to compile it from scratch. You will need:

- Emacs sources. See Section 8.5 [Current GNU distributions], page 50, for a list of ftp sites that make them available. On '`ftp.gnu.org`', the main GNU distribution site, sources are available as

  `ftp://ftp.gnu.org/pub/gnu/emacs/emacs-21.2.tar.gz`

  The above will obviously change as new versions of Emacs come out. For instance, when Emacs 21.42 is released, it will most probably be available as

  `ftp://ftp.gnu.org/pub/gnu/emacs/emacs-21.42.tar.gz`

  Again, you should use one of the GNU mirror sites (see Section 8.5 [Current GNU distributions], page 50, and adjust the URL accordingly) so as to reduce load on '`ftp.gnu.org`'.
- `gzip`, the GNU compression utility. You can get `gzip` via anonymous ftp at mirrors of '`ftp.gnu.org`' sites; it should compile and install without much trouble on most systems. Once you have retrieved the Emacs sources, you will probably be able to uncompress them with the command

  ```
  gunzip --verbose emacs-21.2.tar.gz
  ```

  changing the Emacs version (21.2), as necessary. Once `gunzip` has finished doing its job, a file by the name of '`emacs-21.2.tar`' should be in your build directory.
- `tar`, the *tape archiving* program, which moves multiple files into and out of archive files, or *tarfiles*. All of the files comprising the Emacs source come in a single tarfile, and must be extracted using `tar` before you can build Emacs. Typically, the extraction command would look like

  ```
  tar -xvvf emacs-21.2.tar
  ```

  The '`x`' indicates that we want to extract files from this tarfile, the two '`v`'s force verbose output, and the '`f`' tells `tar` to use a disk file, rather than one on the tape drive.

  If you're using GNU `tar` (available at mirrors of '`ftp.gnu.org`'), you can combine this step and the previous one by using the command

  ```
  tar -zxvvf emacs-21.2.tar.gz
  ```

  The additional '`z`' at the beginning of the options list tells GNU `tar` to uncompress the file with `gunzip` before extracting the tarfile's components.

At this point, the Emacs sources (all 70+ megabytes of them) should be sitting in a directory called '`emacs-21.2`'. On most common Unix and Unix-like systems, you should be able to compile Emacs (with X Window system support) with the following commands:

```
cd emacs-21.2       # change directory to emacs-21.2
./configure         # configure Emacs for your particular system
make                # use Makefile to build components, then Emacs
```

If the `make` completes successfully, the odds are fairly good that the build has gone well. (See Section 7.3 [Problems building Emacs], page 46, if you weren't successful.)

By default, Emacs is installed in the following directories:

'`/usr/local/bin`'
: binaries.

'`/usr/local/share/emacs/21.2`'
: Lisp code and support files.

'`/usr/local/info`'
: Info documentation.

To install files in those default directories, become the superuser and type

```
make install
```

Note that '`make install`' will overwrite '`/usr/local/bin/emacs`' and any Emacs Info files that might be in '`/usr/local/info`'.

Much more verbose instructions (with many more hints and suggestions) come with the Emacs sources, in the file '`INSTALL`'.

7.2 How do I update Emacs to the latest version?

See Section 7.1 [Installing Emacs], page 45, and follow the instructions there for installation.

Most files are placed in version-specific directories. Emacs 21.2, for instance, places files in '`/usr/local/share/emacs/21.2`'.

Upgrading should overwrite only, '`/usr/local/bin/emacs`' (the Emacs binary) and documentation in '`/usr/local/info`'. Back up these files before you upgrade, and you shouldn't have too much trouble.

7.3 What should I do if I have trouble building Emacs?

First look in the file '`etc/PROBLEMS`' (where you unpack the Emacs source) to see if there is already a solution for your problem. Next, look for other questions in this FAQ that have to do with Emacs installation and compilation problems.

If you'd like to have someone look at your problem and help solve it, see Section 3.9 [Help installing Emacs], page 15.

If you cannot find a solution in the documentation, send a message to `bug-gnu-emacs@gnu.org`.

Please don't post it to `news:gnu.emacs.help` or send e-mail to `help-gnu-emacs@gnu.org`. For further guidelines, see Section 2.3 [Guidelines for newsgroup postings], page 7 and Section 2.5 [Reporting bugs], page 8.

7.4 Why does linking Emacs with -lX11 fail?

Emacs needs to be linked with the static version of the X11 library, '`libX11.a`'. This may be missing.

On OpenWindows, you may need to use `add_services` to add the "OpenWindows Programmers" optional software category from the CD-ROM.

On HP-UX 8.0, you may need to run `update` again to load the X11-PRG "fileset". This may be missing even if you specified "all filesets" the first time. If '`libcurses.a`' is missing, you may need to load the "Berkeley Development Option."

David Zuhn (`zoo@armadillo.com`) says that MIT X builds shared libraries by default, and only shared libraries, on those platforms that support them. These shared libraries can't be used when undumping `temacs` (the last stage of the Emacs build process). To get regular libraries in addition to shared libraries, add this to '`site.cf`':

```
#define ForceNormalLib YES
```

Other systems may have similar problems. You can always define `CANNOT_DUMP` and link with the shared libraries instead.

To get the Xmenu stuff to work, you need to find a copy of MIT's '`liboldX.a`'.

8 Finding Emacs and related packages

8.1 Where can I get Emacs on the net (or by snail mail)?

Look in the files 'etc/DISTRIB' and 'etc/FTP' for information on nearby archive sites and 'etc/ORDERS' for mail orders. If you don't already have Emacs, see Section 3.8 [Informational files for Emacs], page 14, for how to get these files.

See Section 7.1 [Installing Emacs], page 45, for information on how to obtain and build the latest version of Emacs, and see Section 8.5 [Current GNU distributions], page 50, for a list of archive sites that make GNU software available.

8.2 How do I find a Emacs Lisp package that does XXX?

First of all, you should check to make sure that the package isn't already available. For example, typing *M-x apropos* ⟨RET⟩ *wordstar* ⟨RET⟩ lists all functions and variables containing the string 'wordstar'.

It is also possible that the package is on your system, but has not been loaded. To see which packages are available for loading, look through your computer's lisp directory (see Section 1.4 [File-name conventions], page 4). The Lisp source to most packages contains a short description of how they should be loaded, invoked, and configured—so before you use or modify a Lisp package, see if the author has provided any hints in the source code.

If a package does not come with Emacs, check the Lisp Code Directory. The LCD was originally maintained by Dave Brennan (brennan@hal.com), but was recently taken over by toby knudsen (toby@world.std.com), who maintains http://www.emacs.org. The LCD is currently being reorganized and updated, but you can meanwhile find many packages at ftp://ftp.emacs.org/pub.

For now, you can search through the LCD with 'lispdir.el', which is in the process of being updated. Download it from the LCD, in the 'emacs-lisp-attic/misc' directory, and then evaluate the following Lisp form (see Section 5.29 [Evaluating Emacs Lisp code], page 29):

```
(setq lisp-code-directory
 "/anonymous@ftp.emacs.org:pub/emacs-lisp-attic/emacs-lisp/LCD-datafile.gz"
  elisp-archive-host "ftp.emacs.org"
  elisp-archive-directory "/pub/emacs-lisp-attic/emacs-lisp/")
```

Once you have installed 'lispdir.el', you can use *M-x lisp-dir-apropos* to search the listing. For example, *M-x lisp-dir-apropos* ⟨RET⟩ *ange-ftp* ⟨RET⟩ produces this output:

```
          GNU Emacs Lisp Code Directory Apropos --- "ange-ftp"
"~/" refers to archive.cis.ohio-state.edu:pub/elisp-archive/

      ange-ftp (4.18)       15-Jul-1992
           Andy Norman, <ange@hplb.hpl.hp.com>
           ~/packages/ange-ftp.tar.Z
           transparent FTP Support for GNU Emacs
      auto-save (1.19)      01-May-1992
           Sebastian Kremer, <sk@thp.uni-koeln.de>
```

```
          ~/misc/auto-save.el.Z
          Safer autosaving with support for ange-ftp and /tmp
     ftp-quik (1.0)       28-Jul-1993
          Terrence Brannon, <tb06@pl122f.eecs.lehigh.edu>
          ~/modes/ftp-quik.el.Z
          Quik access to dired'ing of ange-ftp and normal paths
```

8.3 Where can I get Emacs Lisp packages that don't come with Emacs?

First, check the Lisp Code Directory to find the name of the package you are looking for (see Section 8.2 [Finding a package with particular functionality], page 49). Next, check local archives and the Emacs Lisp Archive to find a copy of the relevant files. If you still haven't found it, you can send e-mail to the author asking for a copy. If you find Emacs Lisp code that doesn't appear in the LCD, please submit a copy to the LCD (see Section 8.4 [Submitting to the Emacs Lisp Archive], page 50).

You can access the Emacs Lisp Archive at

`ftp://archive.cis.ohio-state.edu/pub/emacs-lisp/`

or at

`http://www.cis.ohio-state.edu/emacs-lisp`

Retrieve and read the file '`README`' first.

- The archive maintainers do not have time to answer individual requests for packages or the list of packages in the archive. If you cannot use FTP or UUCP to access the archive yourself, try to find a friend who can, but please don't ask the maintainers.
- Any files with names ending in '`.Z`', '`.z`', or '`.gz`' are compressed, so you should use `"binary"` mode in FTP to retrieve them. You should also use binary mode whenever you retrieve any files with names ending in '`.elc`'.

8.4 How do I submit code to the Emacs Lisp Archive?

Guidelines and procedures for submission to the archive can be found in the file '`GUIDELINES`' in the archive directory (see Section 8.3 [Packages that do not come with Emacs], page 50). It covers documentation, copyrights, packaging, submission, and the Lisp Code Directory Record. Anonymous FTP uploads are not permitted. Instead, all submissions are mailed to `elisp-archive@cis.ohio-state.edu`. The '`lispdir.el`' package has a function named `submit-lcd-entry` which will help you with this.

8.5 Where can I get other up-to-date GNU stuff?

The most up-to-date official GNU software is normally kept at

`ftp://ftp.gnu.org/pub/gnu`

Read the files '`etc/DISTRIB`' and '`etc/FTP`' for more information.

A list of sites mirroring '`ftp.gnu.org`' can be found at

`http://www.gnu.org/order/ftp.html`

8.6 What is the difference between Emacs and XEmacs (formerly "Lucid Emacs")?

First of all, they're both GNU Emacs. XEmacs is just as much a later version of GNU Emacs as the FSF-distributed version. This FAQ refers to the latest version to be distributed by the FSF as "Emacs," partly because the XEmacs maintainers now refer to their product using the "XEmacs" name, and partly because there isn't any accurate way to differentiate between the two without getting mired in paragraphs of legalese and history.

XEmacs, which began life as Lucid Emacs, is based on an early version of Emacs 19 and Epoch, an X-aware version of Emacs 18.

Emacs (i.e., the version distributed by the FSF) has a larger installed base and now always contains the MULE multilingual facilities. XEmacs can do some clever tricks with X and MS-Windows, such as putting arbitrary graphics in a buffer. Similar facilities have been implemented for Emacs as part of a new redisplay implementation for Emacs 21, and are available in the latest Emacs releases. Emacs and XEmacs each come with Lisp packages that are lacking in the other; RMS says that the FSF would include more packages that come with XEmacs, but that the XEmacs maintainers don't always keep track of the authors of contributed code, which makes it impossible for the FSF to have certain legal papers signed. (Without these legal papers, the FSF will not distribute Lisp packages with Emacs.) The two versions have some significant differences at the Lisp programming level.

Many XEmacs features have found their way into recent versions of Emacs, and more features can be expected in the future, but there are still many differences between the two.

The latest stable version of XEmacs as of this writing is 21.1.14; you can get it at

`ftp://ftp.xemacs.org/pub/xemacs/xemacs-21.1/xemacs-21.1.14.tar.gz`

More information about XEmacs, including a list of frequently asked questions (FAQ), is available at

`http://www.xemacs.org/`

8.7 Where can I get Emacs for my PC running MS-DOS?

A pre-built binary distribution of Emacs is available from the SimTel.NET archives. This version apparently works under MS-DOS and Windows (3.X, 9X, ME, NT, and 2000) and supports long file names under Windows 9X, Windows ME, and Windows 2000. More information is available from

`http://www.simtel.net/pub/djgpp/v2gnu/emacs.README`

The binary itself is available in the files '`em*.zip`' in the directory

`http://www.simtel.net/pub/djgpp/v2gnu/`

If you prefer to compile Emacs for yourself, you can do so with the current distribution directly. You will need a 386 (or better) processor, and to be running MS-DOS 3.0 or later. According to Eli Zaretskii (`eliz@is.elta.co.il`) and Darrel Hankerson (`hankedr@dms.auburn.edu`), you will need the following:

Compiler DJGPP version 1.12 maint 1 or later. Djgpp 2.0 or later is recommended, since 1.x is very old an unmaintained. Djgpp 2 supports long file names on Windows 9X/ME/2K.

You can get the latest release of DJGPP by retrieving all of the files in `http://www.simtel.net/pub/djgpp/v2*`

Unpacking program

The easiest way is to use `djtar` which comes with DJGPP v2.x, because it can open gzip'ed tarfiles (i.e., those ending with '`.tar.gz`') in one step. `Djtar` comes in '`djdevnnn.zip`' archive (where *nnn* is the DJGPP version number), from the URL mentioned above.

Warning! Do **not** use the popular WinZip program to unpack the Emacs distribution! WinZip is known to corrupt some of the files by converting them to the DOS CR-LF format, it doesn't always preserve the directory structure recorded in the compressed Emacs archive, and commits other atrocities. Some of these problems could actually prevent Emacs from building successfully!

make, mv, sed, and rm

All of these utilities are available at

`http://www.simtel.net/pub/djgpp/v2gnu`

16-bit utilities can be found in GNUish, at

`http://www.simtel.net/pub/gnuish/`

(`mv` and `rm` are in the Fileutils package, `sed` and `make` are each one in a separate package named after them.)

The files '`INSTALL`' (near its end) and '`etc/PROBLEMS`' in the directory of the Emacs sources contains some additional information regarding Emacs under MS-DOS.

For a list of other MS-DOS implementations of Emacs (and Emacs look-alikes), consult the list of "Emacs implementations and literature," available at

`ftp://rtfm.mit.edu/pub/usenet/comp.emacs/`

Note that while many of these programs look similar to Emacs, they often lack certain features, such as the Emacs Lisp extension language.

8.8 Where can I get Emacs for Microsoft Windows

For information on Emacs for Windows 95 and NT, read the FAQ produced by Geoff Voelker (`voelker@cs.washington.edu`), available at

`http://www.gnu.org/software/emacs/windows/ntemacs.html`

See Section 8.7 [Emacs for MS-DOS], page 51, for Windows 3.1.

A port of Emacs 20.7 for Windows CE, based on NTEmacs, is available at

`http://www.rainer-keuchel.de/software.html`

This port was done by Rainer Keuchel (`coyxc@rainer-keuchel.de`), and supports all Emacs features except async subprocesses and menus. You will need MSVC 6.0 and a Windows CE SDK to build this port.

8.9 Where can I get Emacs for my PC running OS/2?

Emacs 20.6 is ported for emx on OS/2 2.0 or 2.1, and is available at

`ftp://hobbes.nmsu.edu/pub/os2/apps/editors/emacs/e206*.zip`

and also at

`http://archiv.leo.org/pub/comp/os/os2/leo/gnu/emacs%2d20/`

Instructions for installation, basic setup, and other useful information for OS/2 users of Emacs can be found at

`http://userpage.fu-berlin.de/~oheiabbd/emacs/emacs206-os2.html`

8.10 Where can I get Emacs for my Atari ST?

Roland Schuble reports that Emacs 18.58 running on plain TOS and MiNT is available at `ftp://atari.archive.umich.edu/Editors/Emacs-18-58/1858b-d3.zoo`.

8.11 Where can I get Emacs for my Amiga?

The files you need are available at

`ftp://ftp.wustl.edu/pub/aminet/util/gnu/`

David Gilbert (`dgilbert@gamiga.guelphnet.dweomer.org`) has released a beta version of Emacs 19.25 for the Amiga. You can get the binary at

`ftp://ftp.wustl.edu/pub/aminet/util/gnu/a2.0bEmacs-bin.lha`

8.12 Where can I get Emacs for NeXTSTEP?

Emacs.app is a NeXTSTEP version of Emacs 19.34 which supports colors, menus, and multiple frames. You can get it from

`ftp://next-ftp.peak.org/pub/next/apps/emacs/Emacs_for_NeXTstep.4.20a1.NIHS.b.tar.gz`

8.13 Where can I get Emacs for my Apple computer?

An unofficial port of GNU Emacs 18.59 to the Macintosh is available at a number of ftp sites, the home being `ftp://ftp.cs.cornell.edu/pub/parmet/Emacs-1.17.sit.bin`.

A port of Emacs 20.4 is available at `http://www.cs.hku.hk/~choi/emacs/index.html`.

Beginning with version 21.1, the Macintosh is supported in the official Emacs distribution; see the files '`mac/README`' and '`mac/INSTALL`' in the Emacs distribution for build instructions.

Apple's forthcoming `"OS X"` is based largely on NeXTSTEP and OpenStep. See Section 8.12 [Emacs for NeXTSTEP], page 53, for more details about that version.

8.14 Where do I get Emacs that runs on VMS under DECwindows?

Up-to-date information about GNU software (including Emacs) for VMS is available at `http://vms.gnu.org/`.

8.15 Where can I get modes for Lex, Yacc/Bison, Bourne shell, csh, C++, Objective-C, Pascal, Java, and Awk?

Most of these modes are now available in standard Emacs distribution. To get additional modes, look in the Lisp Code Directory (see Section 8.2 [Finding a package with particular functionality], page 49). For C++, if you use `lisp-dir-apropos`, you must specify the pattern with something like *`M-x lisp-dir-apropos`* ⟨RET⟩ *`c\+\+`* ⟨RET⟩.[1]

Barry Warsaw's `cc-mode` now works for C, C++, Objective-C, and Java code. You can get the latest version from the Emacs Lisp Archive; see Section 8.3 [Packages that do not come with Emacs], page 50 for details. A FAQ for `cc-mode` is available at `http://www.python.org/emacs/cc-mode/`.

8.16 What is the IP address of XXX.YYY.ZZZ?

If you are on a Unix machine, try using the '`nslookup`' command, included in the Berkeley BIND package. For example, to find the IP address of '`ftp.gnu.org`', you would type `nslookup ftp.gnu.org`.

Your computer should then provide the IP address of that machine.

If your site's nameserver is deficient, you can use IP addresses to FTP files. You can get this information by e-mail:

```
To: dns@[134.214.84.25]    (to grasp.insa-lyon.fr)
Body: ip XXX.YYY.ZZZ        (or "help" for more information
                             and options - no quotes)
```

or:

```
To: resolve@[147.31.254.130]         (to laverne.cs.widener.edu)
Body: site XXX.YYY.ZZZ
```

[1] The backslashes in '`\+\+`' are required because *`M-x lisp-dir-apropos`* expects a regular expression as its argument (see Section 5.36 [Using regular expressions], page 30), and '`+`' has a special meaning in regular expressions.

9 Major packages and programs

9.1 VM (View Mail) — another mail reader within Emacs, with MIME support

Author Kyle Jones (`kyle@uunet.uu.net`)

Latest version
6.72

Distribution
`ftp://ftp.wonderworks.com/pub/vm/vm.tar.gz`

Informational newsgroup/mailing list
`news:gnu.emacs.vm.info`
Subscription requests to `info-vm-request@uunet.uu.net`
Submissions to `info-vm@uunet.uu.net`

Bug reports newsgroup/mailing list
`news:gnu.emacs.vm.bug`
Subscription requests to `bug-vm-request@uunet.uu.net`
Submissions to `bug-vm@uunet.uu.net`

VM 6 works with Emacs 20.4, and may cause problems with Emacs 20.3 and below. (But note that many people seem to use Emacs 20.3 with VM 6, without any problems.) Risk-averse users might wish to try VM 5.97, available from the same FTP site (`ftp://ftp.wonderworks.com/pub/vm/`).

9.2 Supercite — mail and news citation package within Emacs

Author Barry Warsaw (`bwarsaw@cen.com`)

Latest version
3.54 (comes bundled with Emacs 20)

Distribution
`http://www.python.org/emacs/supercite.tar.gz`

Mailing list
Subscription requests to `supercite-request@python.org`
Submissions `supercite@python.org`

Superyank is an old version of Supercite.

9.3 Calc — poor man's Mathematica within Emacs

Author Dave Gillespie (`daveg@csvax.cs.caltech.edu`)

Latest version
2.02f

Distribution
`ftp://ftp.gnu.org/pub/gnu/calc/calc-2.02f.tar.gz`

Note that Calc 2.02f needs patching to work with Emacs 21 and later.

Emacs 21.1 and later comes with a package called '`calculator.el`'. It doesn't support all the mathematical wizardry offered by Calc, such as matrices, special functions, and statistics, but is more than adequate as a replacement for `xcalc` and similar programs.

9.4 VIPER — `vi` emulation for Emacs

Since Emacs 19.29, the preferred `vi` emulation in Emacs is VIPER (*M-x viper-mode* ⟨RET⟩), which comes with Emacs. It extends and supersedes VIP (including VIP 4.3) and provides `vi` emulation at several levels, from one that closely follows `vi` to one that departs from `vi` in several significant ways.

For Emacs 19.28 and earlier, the following version of VIP is generally better than the one distributed with Emacs:

Author Aamod Sane (`sane@cs.uiuc.edu`)

Latest version
4.3

Distribution
`ftp://archive.cis.ohio-state.edu/pub/emacs-lisp/old-archive/modes/vip-mode.tar.Z`

9.5 AUC TeX — enhanced LaTeX mode with debugging facilities

Authors Kresten Krab Thorup (`krab@iesd.auc.dk`) and
Per Abrahamsen (`abraham@dina.kvl.dk`)

Latest version
9.9p

Distribution
`ftp://sunsite.auc.dk/packages/auctex/auctex.tar.gz`

Web site `http://sunsite.auc.dk/auctex/`

Mailing list:
Subscription requests to `auc-tex-request@iesd.auc.dk`
Submissions to `auc-tex@iesd.auc.dk`
Development team is at `auc-tex_mgr@iesd.auc.dk`

9.6 BBDB — personal Info Rolodex integrated with mail/news readers

Maintainer
Matt Simmons (`simmonmt@acm.org`)

Latest version
2.00

Distribution
`http://bbdb.sf.net/`

Mailing lists
Subscription requests to `info-bbdb-request@xemacs.org`
Submissions to `info-bbdb@xemacs.org`
Release announcements: `bbdb-announce-request@xemacs.org`

9.7 Ispell — spell checker in C with interface for Emacs

Author Geoff Kuenning (`geoff@itcorp.com`)

Latest version
3.1.20

Distribution
`ftp://ftp.cs.ucla.edu/pub/ispell/ispell-3.1.20.tar.gz`

Web site `http://fmg-www.cs.ucla.edu/geoff/ispell.html`

- Do not ask Geoff to send you the latest version of Ispell. He does not have free e-mail.
- This Ispell program is distinct from GNU Ispell 4.0. GNU Ispell 4.0 is no longer a supported product.

9.8 w3-mode — A World Wide Web browser inside of Emacs

Author Bill Perry (`wmperry@spry.com`)

Latest version
4.0pre.39

Distribution
`ftp://ftp.cs.indiana.edu/pub/elisp/w3/w3.tar.gz`

Mailing lists
Receive announcements from `w3-announce-request@indiana.edu`
Become a beta tester at `w3-beta-request@indiana.edu`
Help to develop `w3-mode` at `w3-dev@indiana.edu`

9.9 EDB — Database program for Emacs; replaces forms editing modes

Author Michael Ernst (mernst@theory.lcs.mit.edu)

Latest version
1.21

Distribution
ftp://theory.lcs.mit.edu/pub/emacs/edb

9.10 Mailcrypt — PGP interface within Emacs mail and news

Authors Patrick J. LoPresti (patl@lcs.mit.edu) and Jin S. Choi (jin@atype.com)

Maintainer
Len Budney (lbudney@pobox.com)

Latest version
3.5.3

Distribution
http://www.nb.net/~lbudney/linux/software/mailcrypt/mailcrypt-3.5.3.tar.gz

Web site http://www.nb.net/~lbudney/linux/software/mailcrypt.html

9.11 JDE — Integrated development environment for Java

Author Paul Kinnucan (paulk@mathworks.com)

Mailing list
jde-subscribe@sunsite.auc.dk

Latest version
2.1.1

Web site http://sunsite.auc.dk/jde/

9.12 Patch — program to apply "diffs" for updating files

Author Larry Wall (lwall@wall.org) (with GNU modifications)

Latest version
2.5.4

Distribution
See Section 8.5 [Current GNU distributions], page 50.

10 Key bindings

10.1 How do I bind keys (including function keys) to commands?

Keys can be bound to commands either interactively or in your '`.emacs`' file. To interactively bind keys for all modes, type *M-x global-set-key* ⟨RET⟩ *key cmd* ⟨RET⟩.

To bind a key just in the current major mode, type *M-x local-set-key* ⟨RET⟩ *key cmd* ⟨RET⟩.

See Info file '`emacs`', node '`Key Bindings`', for further details.

To make the process of binding keys interactively eaiser, use the following "trick": First bind the key interactively, then immediately type *C-x* ⟨ESC⟩ ⟨ESC⟩ *C-a C-k C-g*. Now, the command needed to bind the key is in the kill ring, and can be yanked into your '`.emacs`' file. If the key binding is global, no changes to the command are required. For example,

```
(global-set-key (quote [f1]) (quote help-for-help))
```

can be placed directly into the '`.emacs`' file. If the key binding is local, the command is used in conjunction with the "add-hook" command. For example, in tex-mode, a local binding might be

```
(add-hook 'tex-mode-hook
  (lambda ()
   (local-set-key (quote [f1]) (quote help-for-help))))
```

- Control characters in key sequences, in the form yanked from the kill ring are given in their graphic form—i.e., ⟨CTRL⟩ is shown as '`^`', ⟨TAB⟩ as a set of spaces (usually 8), etc. You may want to convert these into their vector or string forms.
- If a prefix key of the character sequence to be bound is already bound as a complete key, then you must unbind it before the new binding. For example, if *ESC* `{` is previously bound:

  ```
  (global-unset-key [?\e ?{])   ;;   or
  (local-unset-key [?\e ?{])
  ```
- Aside from commands and "lambda lists," a vector or string also can be bound to a key and thus treated as a macro. For example:

  ```
  (global-set-key [f10] [?\C-x?\e?\e?\C-a?\C-k?\C-g])  ;;  or
  (global-set-key [f10] "\C-x\e\e\C-a\C-k\C-g")
  ```

10.2 Why does Emacs say "Key sequence XXX uses invalid prefix characters"?

Usually, one of two things has happened. In one case, the control character in the key sequence has been misspecified (e.g. '`C-f`' used instead of '`\C-f`' within a Lisp expression). In the other case, a *prefix key* in the keystroke sequence you were trying to bind was already bound as a *complete key*. Historically, the '`ESC [`' prefix was usually the problem, in which case you should evaluate either of these forms before attempting to bind the key sequence:

```
(global-unset-key [?\e ?[])  ;;  or
(global-unset-key "\e[")
```

10.3 Why doesn't this [terminal or window-system setup] code work in my '.emacs' file, but it works just fine after Emacs starts up?

During startup, Emacs initializes itself according to a given code/file order. If some of the code executed in your '.emacs' file needs to be postponed until the initial terminal or window-system setup code has been executed but is not, then you will experience this problem (this code/file execution order is not enforced after startup).

To postpone the execution of Emacs Lisp code until after terminal or window-system setup, treat the code as a *lambda list* and set the value of either the `term-setup-hook` or `window-setup-hook` variable to this lambda function. For example,

```
(add-hook 'term-setup-hook
          (lambda ()
           (when (string-match "\\`vt220" (or (getenv "TERM") ""))
             ;; Make vt220's "Do" key behave like M-x:
             (global-set-key [do] 'execute-extended-command))))
```

For information on what Emacs does every time it is started, see the '`lisp/startup.el`' file.

10.4 How do I use function keys under X?

With Emacs 19, functions keys under X are bound like any other key. See Section 10.1 [Binding keys to commands], page 59, for details.

10.5 How do I tell what characters or symbols my function or arrow keys emit?

Type `C-h c` then the function or arrow keys. The command will return either a function key symbol or character sequence (see the Emacs on-line documentation for an explanation). This works for other keys as well.

10.6 How do I set the X key "translations" for Emacs?

Emacs is not written using the Xt library by default, so there are no "translations" to be set. (We aren't sure how to set such translations if you do build Emacs with Xt; please let us know if you've done this!)

The only way to affect the behavior of keys within Emacs is through `xmodmap` (outside Emacs) or `define-key` (inside Emacs). The `define-key` command should be used in conjunction with the `function-key-map` map. For instance,

```
(define-key function-key-map [M-⟨TAB⟩] [?\M-\t])
```

defines the `M-⟨TAB⟩` key sequence.

10.7 How do I handle `C-s` and `C-q` being used for flow control?

`C-s` and `C-q` are used in the XON/XOFF flow control protocol. This messes things up when you're using Emacs over a serial line, because Emacs binds these keys to commands by default. Because Emacs won't honor them as flow control characters, too many of these characters are not passed on and overwhelm output buffers. Sometimes, intermediate software using XON/XOFF flow control will prevent Emacs from ever seeing `C-s` and `C-q`.

Possible solutions:

- Disable the use of `C-s` and `C-q` for flow control.

  You need to determine the cause of the flow control.

  - your terminal

    Your terminal may use XON/XOFF flow control to have time to display all the characters it receives. For example, VT series terminals do this. It may be possible to turn this off from a setup menu. For example, on a VT220 you may select "No XOFF" in the setup menu. This is also true for some terminal emulation programs on PCs.

    When you turn off flow control at the terminal, you will also need to turn it off at the other end, which might be at the computer you are logged in to or at some terminal server in between.

    If you turn off flow control, characters may be lost; using a printer connected to the terminal may fail. You may be able to get around this problem by modifying the '`termcap`' entry for your terminal to include extra NUL padding characters.

  - a modem

    If you are using a dialup connection, the modems may be using XON/XOFF flow control. It's not clear how to get around this.

  - a router or terminal server

    Some network box between the terminal and your computer may be using XON/XOFF flow control. It may be possible to make it use some other kind of flow control. You will probably have to ask your local network experts for help with this.

  - `tty` and/or `pty` devices

    If your connection to Emacs goes through multiple `tty` and/or `pty` devices, they may be using XON/XOFF flow control even when it is not necessary.

    Eirik Fuller (`eirik@theory.tn.cornell.edu`) writes:

    > Some versions of `rlogin` (and possibly `telnet`) do not pass flow control characters to the remote system to which they connect. On such systems, Emacs on the remote system cannot disable flow control on the local system. Sometimes '`rlogin -8`' will avoid this problem.
    >
    > One way to cure this is to disable flow control on the local host (the one running `rlogin`, not the one running `rlogind`) using the `stty` command, before starting the `rlogin` process. On many systems, '`stty start u stop u`' will do this.
    >
    > Some versions of '`tcsh`' will prevent even this from working. One way around this is to start another shell before starting rlogin, and issue the '`stty`' command to disable flow control from that shell.

    Use '`stty -ixon`' instead of '`stty start u stop u`' on some systems.

- Make Emacs speak the XON/XOFF flow control protocol.

  You can make Emacs treat *C-s* and *C-q* as flow control characters by evaluating the form

  ```
  (enable-flow-control)
  ```

  to unconditionally enable flow control or

  ```
  (enable-flow-control-on "vt100" "h19")
  ```

  (using your terminal names instead of '`vt100`' or '`h19`') to enable selectively. These commands will automatically swap *C-s* and *C-q* to *C-* and *C-^*. Variables can be used to change the default swap keys (`flow-control-c-s-replacement` and `flow-control-c-q-replacement`).

  If you are fixing this for yourself, simply put the form in your '`.emacs`' file. If you are fixing this for your entire site, the best place to put it is in the '`site-lisp/site-start.el`' file. (Here '`site-lisp`' is actually a subdirectory of your Emacs installation directory, typically '`/usr/local/share/emacs`'.) Putting this form in '`site-lisp/default.el`' has the problem that if the user's '`.emacs`' file has an error, this will prevent '`default.el`' from being loaded and Emacs may be unusable for the user, even for correcting their '`.emacs`' file (unless they're smart enough to move it to another name).

  `enable-flow-control` can be invoked interactively as well: *M-x enable-flow-control* ⟨RET⟩.

For further discussion of this issue, read the file '`etc/PROBLEMS`' (in the Emacs source directory when you unpack the Emacs distribution).

10.8 How do I bind *C-s* and *C-q* (or any key) if these keys are filtered out?

To bind *C-s* and *C-q*, use either `enable-flow-control` or `enable-flow-control-on`. See Section 10.7 [Handling C-s and C-q with flow control], page 60, for usage and implementation details.

To bind other keys, use `keyboard-translate`. See Section 10.11 [Swapping keys], page 64, for usage details. To do this for an entire site, you should swap the keys in '`site-lisp/site-start.el`'. See Section 10.7 [Handling C-s and C-q with flow control], page 60, for an explanation of why '`site-lisp/default.el`' should not be used.

- If you do this for an entire site, the users will be confused by the disparity between what the documentation says and how Emacs actually behaves.

10.9 Why does the ⟨Backspace⟩ key invoke help?

The ⟨Backspace⟩ key (on most keyboards) generates ASCII code 8. *C-h* sends the same code. In Emacs by default *C-h* invokes help-command. This is intended to be easy to remember since the first letter of '`help`' is '`h`'. The easiest solution to this problem is to use *C-h* (and ⟨Backspace⟩) for help and ⟨DEL⟩ (the ⟨Delete⟩ key) for deleting the previous character.

For many people this solution may be problematic:

- They normally use ⟨Backspace⟩ outside of Emacs for deleting the previous character. This can be solved by making ⟨DEL⟩ the command for deleting the previous character outside of Emacs. On many Unix systems, this command will remap ⟨DEL⟩:

  ```
  stty erase `^?'
  ```

- The user may prefer the ⟨Backspace⟩ key for deleting the previous character because it is more conveniently located on their keyboard or because they don't even have a separate ⟨Delete⟩ key. In this case, the ⟨Backspace⟩ key should be made to behave like ⟨Delete⟩. There are several methods.
  - Some terminals (e.g., VT3## terminals) and terminal emulators (e.g., TeraTerm) allow the character generated by the ⟨Backspace⟩ key to be changed from a setup menu.
  - You may be able to get a keyboard that is completely programmable, or a terminal emulator that supports remapping of any key to any other key.
  - With Emacs 21.1 and later, you can control the effect of the ⟨Backspace⟩ and ⟨Delete⟩ keys, on both dumb terminals and a windowed displays, by customizing the option `normal-erase-is-backspace-mode`, or by invoking *`M-x normal-erase-is-backspace`*. See the documentation of these symbols (see Section 3.4 [Emacs Lisp documentation], page 12) for more info.
  - It is possible to swap the ⟨Backspace⟩ and ⟨DEL⟩ keys inside Emacs:

    ```
    (keyboard-translate ?\C-h ?\C-?)
    ```

    This is the recommended method of forcing ⟨Backspace⟩ to act as ⟨DEL⟩, because it works even in modes which bind ⟨DEL⟩ to something other than `delete-backward-char`.

    Similarly, you could remap ⟨DEL⟩ to act as *`C-d`*, which by default deletes forward:

    ```
    (keyboard-translate ?\C-? ?\C-d)
    ```

    See Section 10.11 [Swapping keys], page 64, for further details about `keyboard-translate`.
  - Another approach is to switch key bindings and put help on *`C-x h`* instead:

    ```
    (global-set-key "\C-h" 'delete-backward-char)

    ;;; overrides mark-whole-buffer
    (global-set-key "\C-xh" 'help-command)
    ```

    This method is not recommended, though: it only solves the problem for those modes which bind ⟨DEL⟩ to `delete-backward-char`. Modes which bind ⟨DEL⟩ to something else, such as `view-mode`, will not work as you expect when you press the ⟨Backspace⟩ key. For this reason, we recommend the the `keyboard-translate` method, shown above.

    Other popular key bindings for help are *`M-?`* and *`C-x ?`*.

  Don't try to bind ⟨DEL⟩ to `help-command`, because there are many modes that have local bindings of ⟨DEL⟩ that will interfere.

When Emacs 21 or later runs on a windowed display, it binds the ⟨Delete⟩ key to a command which deletes the character at point, to make Emacs more consistent with keyboard operation on these systems.

For more information about troubleshooting this problem, see section "If ⟨DEL⟩ Fails to Delete" in *The GNU Emacs Manual*.

10.10 Why doesn't Emacs look at the '`stty`' settings for ⟨Backspace⟩ vs. ⟨Delete⟩?

Good question!

10.11 How do I swap two keys?

In Emacs 19, you can swap two keys (or key sequences) by using the `keyboard-translate` function. For example, to turn *C-h* into ⟨DEL⟩ and ⟨DEL⟩ to *C-h*, use

```
(keyboard-translate ?\C-h ?\C-?)  ; translate 'C-h' to DEL
(keyboard-translate ?\C-? ?\C-h)  ; translate DEL to 'C-h'.
```

The first key sequence of the pair after the function identifies what is produced by the keyboard; the second, what is matched for in the keymaps.

Keyboard translations are not the same as key bindings in keymaps. Emacs contains numerous keymaps that apply in different situations, but there is only one set of keyboard translations, and it applies to every character that Emacs reads from the terminal. Keyboard translations take place at the lowest level of input processing; the keys that are looked up in keymaps contain the characters that result from keyboard translation.

See Info file '`emacs`', node '`Keyboard Translations`'.

10.12 How do I produce C-XXX with my keyboard?

On terminals (but not under X), some common "aliases" are:

C-2 or *C-*⟨SPC⟩
: *C-@*

C-6
: *C-^*

C-7 or *C-S--*
: *C-_*

C-4
: *C-*

C-5
: *C-]*

C-/
: *C-?*

Often other aliases exist; use the *C-h c* command and try ⟨CTRL⟩ with all of the digits on your keyboard to see what gets generated. You can also try the *C-h w* command if you know the name of the command.

10.13 What if I don't have a ⟨Meta⟩ key?

On many keyboards, the ⟨Alt⟩ key acts as ⟨Meta⟩, so try it.

Instead of typing `M-a`, you can type ⟨ESC⟩ `a`. In fact, Emacs converts `M-a` internally into ⟨ESC⟩ `a` anyway (depending on the value of `meta-prefix-char`). Note that you press ⟨Meta⟩ and ⟨a⟩ together, but with ⟨ESC⟩, you press ⟨ESC⟩, release it, and then press ⟨a⟩.

10.14 What if I don't have an ⟨Escape⟩ key?

Type *C-[* instead. This should send ASCII code 27 just like an Escape key would. *C-3* may also work on some terminal (but not under X). For many terminals (notably DEC terminals) ⟨F11⟩ generates ⟨ESC⟩. If not, the following form can be used to bind it:

```
;;; F11 is the documented ESC replacement on DEC terminals.
(define-key function-key-map [f11] [?\e])
```

10.15 Can I make my ⟨Compose Character⟩ key behave like a ⟨Meta⟩ key?

On a dumb terminal such as a VT220, no. It is rumored that certain VT220 clones could have their ⟨Compose⟩ key configured this way. If you're using X, you might be able to do this with the `xmodmap` command.

10.16 How do I bind a combination of modifier key and function key?

With Emacs 19 and later, you can represent modified function keys in vector format by adding prefixes to the function key symbol. For example (from the on-line documentation):

```
(global-set-key [?\C-x right] 'forward-page)
```

where '`?\C-x`' is the Lisp character constant for the character *C-x*.

You can use the modifier keys ⟨Control⟩, ⟨Meta⟩, ⟨Hyper⟩, ⟨Super⟩, ⟨Alt⟩, and ⟨Shift⟩ with function keys. To represent these modifiers, prepend the strings '`C-`', '`M-`', '`H-`', '`s-`', '`A-`', and '`S-`' to the symbol name. Here is how to make *H-M-RIGHT* move forward a word:

```
(global-set-key [H-M-right] 'forward-word)
```

- Not all modifiers are permitted in all situations. ⟨Hyper⟩, ⟨Super⟩, and ⟨Alt⟩ are not available on Unix character terminals. Non-ASCII keys and mouse events (e.g. *C-=* and *Mouse-1*) also fall under this category.

See Section 10.1 [Binding keys to commands], page 59, for general key binding instructions.

10.17 Why doesn't my ⟨Meta⟩ key work in an `xterm` window?

See Info file '`emacs`', node '`Single-Byte Character Support`'.

If the advice in the Emacs manual fails, try all of these methods before asking for further help:

- You may have big problems using `mwm` as your window manager. (Does anyone know a good generic solution to allow the use of the ⟨Meta⟩ key in Emacs with '`mwm`'?)
- For X11: Make sure it really is a ⟨Meta⟩ key. Use `xev` to find out what keysym your ⟨Meta⟩ key generates. It should be either `Meta_L` or `Meta_R`. If it isn't, use '`xmodmap`' to fix the situation. If ⟨Meta⟩ does generate `Meta_L` or `Meta_R`, but *M-x* produces a non-ASCII character, put this in your '`~/.Xdefaults`' file:

  ```
  XTerm*eightBitInput:   false
  XTerm*eightBitOutput:  true
  ```

- Make sure the `pty` the `xterm` is using is passing 8 bit characters. '`stty -a`' (or '`stty everything`') should show '`cs8`' somewhere. If it shows '`cs7`' instead, use '`stty cs8 -istrip`' (or '`stty pass8`') to fix it.
- If there is an `rlogin` connection between `xterm` and Emacs, the '`-8`' argument may need to be given to rlogin to make it pass all 8 bits of every character.
- If Emacs is running on Ultrix, it is reported that evaluating `(set-input-mode t nil)` helps.
- If all else fails, you can make `xterm` generate ⟨ESC⟩ *W* when you type *M-W*, which is the same conversion Emacs would make if it got the *M-W* anyway. In X11R4, the following resource specification will do this:

  ```
  XTerm.VT100.EightBitInput: false
  ```

  (This changes the behavior of the `insert-eight-bit` action.)

  With older `xterms`, you can specify this behavior with a translation:

  ```
  XTerm.VT100.Translations: #override \
    Meta<KeyPress>: string(0x1b) insert()
  ```

  You might have to replace '`Meta`' with '`Alt`'.

10.18 Why doesn't my ⟨ExtendChar⟩ key work as a ⟨Meta⟩ key under HP-UX 8.0 and 9.x?

This is a result of an internationalization extension in X11R4 and the fact that HP is now using this extension. Emacs assumes that the `XLookupString` function returns the same result regardless of the ⟨Meta⟩ key state which is no longer necessarily true. Until Emacs is fixed, the temporary kludge is to run this command after each time the X server is started but preferably before any xterm clients are:

```
xmodmap -e 'remove mod1 = Mode_switch'
```

This will disable the use of the extra keysyms systemwide, which may be undesirable if you actually intend to use them.

11 Alternate character sets

11.1 How do I make Emacs display 8-bit characters?

See Info file '`emacs`', node '`Single-Byte Character Support`'. On a Unix, when Emacs runs on a text-only terminal display or is invoked with '`emacs -nw`', you typically need to use `set-terminal-coding-system` to tell Emacs what the terminal can display, even after setting the language environment; otherwise non-ASCII characters will display as '`?`'. On other operating systems, such as MS-DOS and MS-Windows, Emacs queries the OS about the character set supported by the display, and sets up the required terminal coding system automatically.

11.2 How do I input eight-bit characters?

Various methods are available for input of eight-bit characters. See See Info file '`emacs`', node '`Single-Byte Character Support`'. For more sophisticated methods, See Info file '`emacs`', node '`Input Methods`'.

11.3 Where can I get an Emacs that handles kanji, Chinese, or other Far-Eastern character sets?

Emacs 20 and later includes many of the features of MULE, the MULtilingual Enhancement to Emacs. See Section 7.1 [Installing Emacs], page 45, for information on where to find and download the latest version of Emacs.

11.4 Where is an Emacs that can handle Semitic (right-to-left) alphabets?

Emacs 20 and later supports Hebrew characters (ISO 8859-8), but does not yet support right-to-left character entry and display.

Joel M. Hoffman (`joel@exc.com`) has written a Lisp package called '`hebrew.el`' that allows right-to-left editing of Hebrew. It reportedly works out of the box with Emacs 19, but requires patches for Emacs 18. Write to Joel if you want the patches or package.

'`hebrew.el`' requires a Hebrew screen font, but no other hardware support. Joel has a screen font for PCs running MS-DOS or GNU/Linux.

You might also try to query archie for files named with '`hebrew`'; several ftp sites in Israel may also have the necessary files.

12 Mail and news

12.1 How do I change the included text prefix in mail/news followups?

If you read mail with Rmail or news with Gnus, set the variable `mail-yank-prefix`. For VM, set `vm-included-text-prefix`. For mh-e, set `mh-ins-buf-prefix`.

For fancier control of citations, use Supercite. See Section 9.2 [Supercite], page 55.

To prevent Emacs from including various headers of the replied-to message, set the value of `mail-yank-ignored-headers` to an appropriate regexp.

12.2 How do I save a copy of outgoing mail?

You can either mail yourself a copy by including a '`BCC`' header in the mail message, or store a copy of the message directly to a file by including an '`FCC`' header.

If you use standard mail, you can automatically create a '`BCC`' to yourself by putting

```
(setq mail-self-blind t)
```

in your '`.emacs`' file. You can automatically include an '`FCC`' field by putting something like the following in your '`.emacs`' file:

```
(setq mail-archive-file-name (expand-file-name "~/outgoing"))
```

The output file will be in Unix mail format, which can be read directly by VM, but not always by Rmail. See Section 3.2 [Learning how to do something], page 11.

If you use `mh-e`, add an '`FCC`' or '`BCC`' field to your components file.

It does not work to put '`set record filename`' in the '`.mailrc`' file.

12.3 Why doesn't Emacs expand my aliases when sending mail?

- You must separate multiple addresses in the headers of the mail buffer with commas. This is because Emacs supports RFC822 standard addresses like this one:

  ```
  To: Willy Smith <wks@xpnsv.lwyrs.com>
  ```

  However, you do not need to—and probably should not, unless your system's version of '`/usr/ucb/mail`' (a.k.a. `mailx`) supports RFC822—separate addresses with commas in your '`~/.mailrc`' file.
- Emacs normally only reads the '`.mailrc`' file once per session, when you start to compose your first mail message. If you edit '`.mailrc`', you can type *`M-x rebuild-mail-abbrevs`* ⟨RET⟩ to make Emacs reread '`~/.mailrc`'.
- If you like, you can expand mail aliases as abbrevs, as soon as you type them in. To enable this feature, execute the following:

  ```
  (add-hook 'mail-mode-hook 'mail-abbrevs-setup)
  ```

  Note that the aliases are expanded automatically only after you type ⟨RET⟩ or a punctuation character (e.g. `,`). You can force their expansion by moving point to the end of the alias and typing *`C-x a e`* (*`M-x expand-abbrev`*).

12.4 Why does Rmail think all my saved messages are one big message?

A file created through the '`FCC`' field in a message is in Unix mail format, not the format that Rmail uses (BABYL format). Rmail will try to convert a Unix mail file into BABYL format on input, but sometimes it makes errors. For guaranteed safety, you can make the '`saved-messages`' file be an inbox for your Rmail file by using the function `set-rmail-inbox-list`.

12.5 How can I sort the messages in my Rmail folder?

In Rmail, type *C-c C-s C-h* to get a list of sorting functions and their key bindings.

12.6 Why does Rmail need to write to '`/usr/spool/mail`'?

This is the behavior of the `movemail` program which Rmail uses. This indicates that `movemail` is configured to use lock files.

RMS writes:

> Certain systems require lock files to interlock access to mail files. On these systems, `movemail` must write lock files, or you risk losing mail. You simply must arrange to let `movemail` write them.
>
> Other systems use the `flock` system call to interlock access. On these systems, you should configure `movemail` to use `flock`.

12.7 How do I recover my mail files after Rmail munges their format?

If you have just done *M-x rmail-input* on a file and you don't want to save it in Rmail's format (called BABYL), just kill the buffer (with *C-x k*).

If you typed *M-x rmail* and it read some messages out of your inbox and you want to put them in a Unix mail file, use *C-o* on each message.

If you want to convert an existing file from BABYL format to Unix mail format, use the command *M-x unrmail*: it will prompt you for the input and output file names.

Alternatively, you could use the `b2m` program supplied with Emacs. `b2m` is a filter, and is used like this:

    b2m < babyl-file > mbox-file

where *babyl-file* is the name of the BABYL file, and *mbox-file* is the name of the file where the converted mail will be written.

12.8 How can I force Rmail to reply to the sender of a message, but not the other recipients?

Ron Isaacson (`isaacson@seas.upenn.edu`) says: When you hit r to reply in Rmail, by default it CCs all of the original recipients (everyone on the original '`To`' and '`CC`' lists). With a prefix argument (i.e., typing *C-u* before r), it replies only to the sender. However, going through the whole *C-u* business every time you want to reply is a pain. This is the best fix I've been able to come up with:

```
(defun rmail-reply-t ()
  "Reply only to the sender of the current message. (See rmail-reply.)"
  (interactive)
  (rmail-reply t))

(add-hook 'rmail-mode-hook
  (lambda ()
    (define-key rmail-mode-map "r" 'rmail-reply-t)
    (define-key rmail-mode-map "R" 'rmail-reply)))
```

12.9 How can I get my favorite Emacs mail package to support MIME?

Read the Emacs MIME FAQ, maintained by MacDonald Hall Jackson (`trey@cs.berkeley.edu`) at

`http://bmrc.berkeley.edu/~trey/emacs/mime.html`

Version 6.x of VM supports MIME. See Section 9.1 [VM], page 55. Gnus supports MIME in mail and news messages as of version 5.8.1 (Pterodactyl). Rmail has limited support for single-part MIME messages beginning with Emacs 20.3.

12.10 How do I make Emacs automatically start my mail/news reader?

To start Emacs in Gnus:

```
emacs -f gnus
```

in Rmail:

```
emacs -f rmail
```

A more convenient way to start with Gnus:

```
alias gnus 'emacs -f gnus'
gnus
```

It is probably unwise to automatically start your mail or news reader from your '`.emacs`' file. This would cause problems if you needed to run two copies of Emacs at the same time. Also, this would make it difficult for you to start Emacs quickly when you needed to.

12.11 How do I read news under Emacs?

Use *M-x gnus*. It is documented in Info (see Section 3.2 [Learning how to do something], page 11).

12.12 Why doesn't Gnus work via NNTP?

There is a bug in NNTP version 1.5.10, such that when multiple requests are sent to the NNTP server, the server only handles the first one before blocking waiting for more input which never comes. NNTP version 1.5.11 claims to fix this.

You can work around the bug inside Emacs like this:

```
(setq nntp-maximum-request 1)
```

You can find out what version of NNTP your news server is running by telnetting to the NNTP port (usually 119) on the news server machine (i.e., `telnet server-machine 119`). The server should give its version number in the welcome message. Type `quit` to get out.

See Section 6.6 [Spontaneous entry into isearch-mode], page 39, for some additional ideas.

12.13 How do I view news articles with embedded underlining (e.g., ClariNews)?

Underlining appears like this:

```
_^Hu_^Hn_^Hd_^He_^Hr_^Hl_^Hi_^Hn_^Hi_^Hn_^Hg
```

Per Abrahamsen (`abraham@dina.kvl.dk`) suggests using the following code, which uses the underline face to turn such text into true underlining, inconjunction with Gnus:

```
(defun gnus-article-prepare-overstrike ()
  ;; Prepare article for overstrike commands.
  (save-excursion
    (set-buffer gnus-article-buffer)
    (let ((buffer-read-only nil))
    (goto-char (point-min))
      (while (search-forward "\b" nil t)
        (let ((next (following-char))
              (previous (char-after (- (point) 2))))
          (cond ((eq next previous)
                 (delete-region (- (point) 2) (point))
                 (put-text-property (point) (1+ (point))
                                    'face 'bold))
                ((eq next ?_)
                 (delete-region (1- (point)) (1+ (point)))
                 (put-text-property (1- (point)) (point)
                                    'face 'underline))
                ((eq previous ?_)
                 (delete-region (- (point) 2) (point))
                 (put-text-property (point) (1+ (point))
                                    'face 'underline))))))))

(add-hook 'gnus-article-prepare-hook 'gnus-article-prepare-overstrike)
```

Latest versions of Gnus do such a conversion automatically.

If you prefer to do away with underlining altogether, you can destructively remove it with `M-x ununderline-region`; do this automatically via

```
(add-hook 'gnus-article-prepare-hook
  (lambda () (ununderline-region (point-min) (point-max))))
```

12.14 How do I save all the items of a multi-part posting in Gnus?

Use `gnus-uu`. Type *C-c C-v C-h* in the Gnus summary buffer to see a list of available commands.

12.15 How do I make Gnus start up faster?

From the Gnus FAQ (see Section 12.21 [Learning more about Gnus], page 74):

> Pranav Kumar Tiwari (`pktiwari@eos.ncsu.edu`) writes: I posted the same query recently and I got an answer to it. I am going to repeat the answer. What you need is a newer version of gnus, version 5.0.4+. I am using 5.0.12 and it works fine with me with the following settings:

```
(setq gnus-check-new-newsgroups nil
      gnus-read-active-file 'some
      gnus-nov-is-evil nil
      gnus-select-method '(nntp gnus-nntp-server))
```

12.16 How do I catch up all newsgroups in Gnus?

In the '`*Newsgroup*`' buffer, type *M-< C-x (c y C-x) M-0 C-x e*

Leave off the initial *M-<* if you only want to catch up from point to the end of the '`*Newsgroup*`' buffer.

12.17 Why can't I kill in Gnus based on the Newsgroups/Keywords/Control headers?

Gnus will complain that the '`Newsgroups`', '`Keywords`', and '`Control`' headers are "Unknown header" fields.

For the '`Newsgroups`' header, there is an easy workaround: kill on the '`Xref`' header instead, which will be present on any cross-posted article (as long as your site carries the cross-post group).

If you really want to kill on one of these headers, you can do it like this:

```
(gnus-kill nil "^Newsgroups: .*\\(bad\\.group\\|worse\\.group\\)")
```

12.18 How do I get rid of flashing messages in Gnus for slow connections?

Set `nntp-debug-read` to `nil`.

12.19 Why is catch up slow in Gnus?

Because Gnus is marking crosspostings read. You can control this with the variable `gnus-use-cross-reference`.

12.20 Why does Gnus hang for a long time when posting?

David Lawrence (`tale@uunet.uu.net`) explains:

> The problem is almost always interaction between NNTP and C News. NNTP POST asks C News's `inews` to not background itself but rather hang around and give its exit status so it knows whether the post was successful. (That wait will on some systems not return the exit status of the waited for job is a different sort of problem.) It ends up taking a long time because `inews` is calling `relaynews`, which often waits for another `relaynews` to free the lock on the news system so it can file the article.
>
> My preferred solution is to change `inews` to not call `relaynews`, but rather use `newsspool`. This loses some error-catching functionality, but is for the most part safe as `inews` will detect a lot of the errors on its own. The C News folks have sped up `inews`, too, so speed should look better to most folks as that update propagates around.

12.21 Where can I find out more about Gnus?

Look for the Gnus FAQ, available at

`http://www.ccs.neu.edu/software/contrib/gnus/`

Concept Index

#

$

-

.

/

A

B

C

D

E

F

G

H

I

S

T

U

Table of Contents