XEmacs FAQ

Frequently asked questions about XEmacs Last Modified: $Date: 2003/10/10 12:39:27 $

Tony Rossini <rossini@biostat.washington.edu>
Ben Wing <ben@xemacs.org>
Chuck Thompson <cthomp@xemacs.org>
Steve Baur <steve@xemacs.org>
Andreas Kaempf <andreas@sccon.com>
Christian Nybø <chr@mediascience.no>
Sandra Wambold <wambold@xemacs.org>

XEmacs FAQ

This is the guide to the XEmacs Frequently Asked Questions list—a compendium of questions and answers pertaining to one of the finest programs ever written. XEmacs is much more than just a Text Editor.

This FAQ is freely redistributable. This FAQ is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If you have a Web browser, the official hypertext version is at
`http://www.xemacs.org/FAQ/xemacs-faq.html`

1 Introduction, Policy, Credits

Learning XEmacs is a lifelong activity. Even people who have used Emacs for years keep discovering new features. Therefore this document cannot be complete. Instead it is aimed at the person who is either considering XEmacs for their own use, or has just obtained it and is wondering what to do next. It is also useful as a reference to available resources.

The previous maintainer of the FAQ was Anthony Rossini, who started it, after getting tired of hearing JWZ complain about repeatedly having to answer questions. Ben Wing and Chuck Thompson, the principal authors of XEmacs, then took over and Ben did a massive update reorganizing the whole thing. At which point Anthony took back over, but then had to give it up again. Some of the other contributors to this FAQ are listed later in this document.

The previous version was converted to hypertext format, and edited by Steven L. Baur. It was converted back to texinfo by Hrvoje Niksic. The FAQ was then maintained by Andreas Kaempf, who passed it on to ChristianNybø.

If you notice any errors or items which should be added or amended to this FAQ please send email to Sandra Wambold. Include '`XEmacs FAQ`' on the Subject: line.

1.0: Introduction

Q1.0.1: What is XEmacs?

XEmacs is a powerful, highly customizable open source text editor and application development system, with full GUI support. It is protected under the GNU Public License and related to other versions of Emacs, in particular GNU Emacs. Its emphasis is on modern graphical user interface support and an open software development model, similar to Linux. XEmacs has an active development community numbering in the hundreds (and thousands of active beta testers on top of this), and runs on all versions of MS Windows, on Linux, and on nearly every other version of Unix in existence. Support for XEmacs has been supplied by Sun Microsystems, University of Illinois, Lucid, ETL/Electrotechnical Laboratory, Amdahl Corporation, BeOpen, and others, as well as the unpaid time of a great number of individual developers.

Q1.0.2: What is the current version of XEmacs?

XEmacs versions 21.1.* are releases made from the current stable sources. XEmacs versions 21.2.* are releases made from the development sources. Check at `http://www.xemacs.org` for the current minor version.

XEmacs 19.16 was the last release of v19, released in November, 1997, which was also the last version without international language support.

Q1.0.3: Where can I find it?

The canonical source and binaries can be found via anonymous FTP at:

`ftp://ftp.xemacs.org/pub/xemacs/`

Q1.0.4: Why Another Version of Emacs?

For a detailed description of the differences between GNU Emacs and XEmacs and a detailed history of XEmacs, check out the

`NEWS file`

However, here is a list of some of the reasons why we think you might consider using it:

- It looks nicer.
- The XEmacs maintainers are generally more receptive to suggestions than the GNU Emacs maintainers.
- Many more bundled packages than GNU Emacs.
- Binaries are available for many common operating systems.
- Face support on TTY's.
- A built-in toolbar.
- Better Motif compliance.
- Some internationalization support (including full MULE support, if compiled with it).
- Variable-width fonts.
- Variable-height lines.
- Marginal annotations.
- ToolTalk support.
- XEmacs can be used as an Xt widget, and can be embedded within another application.
- Horizontal and vertical scrollbars (using real toolkit scrollbars).
- Better APIs (and performance) for attaching fonts, colors, and other properties to text.
- The ability to embed arbitrary graphics in a buffer.
- Completely compatible (at the C level) with the Xt-based toolkits.

Q1.0.5: Why Haven't XEmacs and GNU Emacs Merged?

There are currently irreconcilable differences in the views about technical, programming, design and organizational matters between RMS and the XEmacs development team which provide little hope for a merge to take place in the short-term future.

If you have a comment to add regarding the merge, it is a good idea to avoid posting to the newsgroups, because of the very heated flamewars that often result. Mail your questions to `xemacs-beta@xemacs.org` and `bug-gnu-emacs@prep.ai.mit.edu`.

Q1.0.6: Where can I get help?

Probably the easiest way, if everything is installed, is to use Info, by pressing C-h i , or looking for an Info item on the Help Menu. M-x apropos can be used to look for particular commands.

For items not found in the manual, try reading this FAQ and reading the Usenet group comp.emacs.xemacs.

If you choose to post to a newsgroup, please use comp.emacs.xemacsPlease do not post XEmacs related questions to gnu.emacs.help.

If you cannot post or read Usenet news, there is a corresponding mailing list `xemacs-news@xemacs.org` which is available. It can be subscribed to via the Mailman Web interface or by sending mail to to `xemacs-news-request@xemacs.org` with '`subscribe`' in the body of the message. See also `http://www.xemacs.org/Lists/#xemacs-news`. To cancel a subscription, you may use the `xemacs-news-request@xemacs.org` address or the Web interface. Send a message with a subject of '`unsubscribe`' to be removed.

Q1.0.7: Where are the mailing lists archived?

The archives can be found at `http://list-archive.xemacs.org`

Q1.0.8: How do you pronounce XEmacs?

The most common pronounciation is '`Eks eemax`'.

Q1.0.9: What does XEmacs look like?

Screen snapshots are available at `http://www.xemacs.org/About/Screenshots/index.html` as part of the XEmacs website.

Q1.0.10: Is there a port of XEmacs to Microsoft ('95 or NT)?

Yes. XEmacs can be built under MS Windows and is fully-featured and actively developed. See [MS Windows], page 74.

Q1.0.11: Is there a port of XEmacs to the Macintosh?

Yes.

XEmacs 21.5 (perhaps 21.4 also?) works on MacOS X, although it certainly will not feel very much like a Mac application as it has no Mac-specific code in it.

There is also a port of XEmacs 19.14 that works on all recent versions of MacOS, from 8.1 through MacOS X, by Pitts Jarvis. It runs in an equivalent of TTY mode only (one single Macintosh window, 25 colors), but has a large number of Mac-specific additions. It's available at `http://homepage.mac.com/pjarvis/xemacs.html`.

Q1.0.12: Is there a port of XEmacs to NextStep?

Carl Edman, apparently no longer at `cedman@princeton.edu`, did the port of GNU Emacs to NeXTstep and expressed interest in doing the XEmacs port, but never went any farther.

Q1.0.13: Is there a port of XEmacs to OS/2?

No, but Alexander Nikolaev <avn_1251@mail.ru> is working on it.

Q1.0.14: Where can I obtain a printed copy of the XEmacs User's Manual?

Pre-printed manuals are not available. If you are familiar with TeX, you can generate your own manual from the XEmacs sources.

HTML and Postscript versions of XEmacs manuals are available from the XEmacs web site at `http://www.xemacs.org/Documentation/index.html`.

1.1: Policies

Q1.1.1: What is the FAQ editorial policy?

The FAQ is actively maintained and modified regularly. All links should be up to date. Unfortunately, some of the information is out of date – a situation which the FAQ maintainer is working on. All submissions are welcome, please e-mail submissions to XEmacs FAQ maintainers.

Please make sure that '`XEmacs FAQ`' appears on the Subject: line. If you think you have a better way of answering a question, or think a question should be included, we'd like to hear about it. Questions and answers included into the FAQ will be edited for spelling and grammar and will be attributed. Answers appearing without attribution are either from versions of the FAQ dated before May 1996 or are from previous FAQ maintainers. Answers quoted from Usenet news articles will always be attributed, regardless of the author.

Q1.1.2: How do I become a Beta Tester?

Send an email message to `xemacs-beta-request@xemacs.org` with the line '`subscribe`' in the body of the message.

Be prepared to get your hands dirty, as beta testers are expected to identify problems as best they can.

Q1.1.3: How do I contribute to XEmacs itself?

Ben Wing `ben@xemacs.org` writes:

> BTW if you have a wish list of things that you want added, you have to speak up about it! More specifically, you can do the following if you want a feature added (in increasing order of usefulness):
>
> - Make a posting about a feature you want added.

- Become a beta tester and make more postings about those same features.
- Convince us that you're going to use the features in some cool and useful way.
- Come up with a clear and well-thought-out API concerning the features.
- Write the code to implement a feature and send us a patch.

(not that we're necessarily requiring you to write the code, but we can always hope :)

1.2: Credits

Q1.2.1: Who wrote XEmacs?

XEmacs is the result of the time and effort of many people. The developers responsible for recent releases are:

- Martin Buchholz
- Stephen Turnbull
- Ben Wing
- Hrvoje Niksic

The developers responsible for older releases were:

- Steve Baur
- Chuck Thompson
- Jamie Zawinski
- Richard Mlynarik

  Steve Baur was the primary maintainer for 19.15 through 21.0.

  Chuck Thompson and Ben Wing were the maintainers for 19.11 through 19.14 and heavy code contributors for 19.8 through 19.10.

  Jamie Zawinski was the maintainer for 19.0 through 19.10 (the entire history of Lucid Emacs). Richard Mlynarik was a heavy code contributor to 19.6 through 19.8.

Along with many other contributors, partially enumerated in the '`About XEmacs`' option in the Help menu.

Q1.2.2: Who contributed to this version of the FAQ?

The following people contributed valuable suggestions to building this version of the FAQ (listed in alphabetical order):

- SL Baur
- Hrvoje Niksic
- Aki Vehtari

Q1.2.3: Who contributed to the FAQ in the past?

This is only a partial list, as many names were lost in a hard disk crash some time ago.

- Curtis.N.Bingham
- Georges Brun-Cottan
- Richard Caley
- Richard Cognot
- Mark Daku
- William G. Dubuque
- Eric Eide
- Alain Fauconnet
- Chris Flatters
- Evelyn Ginsparg
- Marty Hall
- Darrell Kindred
- David Moore
- Arup Mukherjee
- Juergen Nickelsen
- Kevin R. Powell
- Justin Sheehy
- Stig
- Aki Vehtari

1.3: Internationalization

Q1.3.1: What is the status of internationalization support aka MULE (including Asian language support?

Both the stable and development versions of XEmacs include internationalization support (aka MULE). MULE currently (21.4) works on UNIX and Linux systems. It is possible to build with MULE on Windows systems, but if you really need MULE on Windows, it is recommended that you build and use the development (21.5) version, and deal with the instability of the development tree. Binaries compiled without MULE support run faster than MULE capable XEmacsen.

Q1.3.2: How can I help with internationalization?

If you would like to help, you may want to join the `xemacs-mule@xemacs.org` mailing list. Especially needed are people who speak/write languages other than English, who are willing to use XEmacs/MULE regularly, and have some experience with Elisp.

Translations of the TUTORIAL and man page are welcome, and XEmacs does support multilingual menus, but we have few current translations.

See [Q1.1.2], page 4.

Q1.3.3: How do I type non-ASCII characters?

See question 3.5.7 (see [Q3.5.7], page 38) in part 3 of this FAQ for some simple methods that also work in non-MULE builds of XEmacs (but only for one-octet coded character sets, and mostly for ISO 8859/1). Many of the methods available for Cyrillic (see [Q1.3.7], page 9) work without MULE. MULE has more general capabilities. See [Q1.3.5], page 7.

See [Q3.2.7], page 34, which covers display of non-ASCII characters.

Q1.3.4: Can XEmacs messages come out in a different language?

The message-catalog support was written but is badly bit-rotted. XEmacs 20 and 21 did *not* support it, and early releases of XEmacs 22 will not either.

However, menubar localization *does* work. To enable it, add to your '`Emacs`' file entries like this:

```
Emacs*XlwMenu.resourceLabels:                    True
Emacs*XlwMenu.file.labelString:                  Fichier
Emacs*XlwMenu.openInOtherWindow.labelString:   In anderem Fenster oeffnen
```

The name of the resource is derived from the non-localized entry by removing punctuation and capitalizing as above.

Q1.3.5: Please explain the various input methods in MULE/XEmacs

Mule supports a wide variety of input methods. There are three basic classes: Lisp implementations, generic platform support, and library interfaces.

Lisp implementations include Quail, which provides table-driven input methods for almost all the character sets that Mule supports (including all of the ISO 8859 family, the Indic languages, Thai, and so on), and SKK, for Japanese. (SKK also supports an interface to an external "dictionary server" process.) Quail supports both typical "dead-key" methods (eg, in the "latin-1-prefix" method, " a produces , LATIN SMALL LETTER A WITH DIAERESIS), and the complex dictionary-based phonetic methods used for Asian ideographic languages like Chinese.

Lisp implementations can be less powerful (but they are not perceptibly inefficient), and of course are not portable to non-Emacs applications. The incompatibility can be very annoying. On the other hand, they require no special platform support or external libraries, so if you can display the characters, Mule can input them for you and you can edit, anywhere.

Generic platform support is currently limited to the X Input Method (XIM) framework, although support for MSIME (for MS Windows) is planned, and IIIMF (Sun's Internet-Intranet Input Method Framework) support is extremely desirable. XIM is enabled at build time by use of the '`--with-xim`' flag to `configure`. For use of XIM, see your platform documentation. However, normally the input method you use is specified via the '`LANG`' and '`XMODIFIERS`' environment variables.

Of course, input skills are portable across most applications. However, especially in modern GUI systems the habit of using bucky bits has fallen into sad disuse, and many XIM systems are poorly configured for use with Emacs. For example, the kinput2 input manager (a separate process providing an interface between Japanese dictionary servers such as Canna and Wnn, and the application) tends to gobble up keystrokes generating Meta characters. This means that to edit while using an XIM input method, you must toggle the input method off every time you want to use `M-f`. Your mileage may vary.

Library interfaces are most common for Japanese, although Wnn supports Chinese (traditional and simplified) and Korean. There are Chinese and Korean input servers available, but we do not know of any patches for XEmacs to use them directly. You can use them via IM-enabled terminals, by manipulating the terminal coding systems. We describe only the Japanese-oriented systems here. The advantage of these systems is that they are very powerful, and on platforms where they are available there is typically a wide range of applications that support them. Thus your input skills are portable across applications.

Mule provides built-in interfaces to the following input methods: Wnn4, Wnn6, Canna, and SJ3. These can be configured at build time. There are patches available (no URL, sorry) to support the SKK server, as well. Wnn and SJ3 use the `egg` user interface. The interface for Canna is specialized to Canna.

Wnn supports Japanese, Chinese and Korean. It is made by OMRON and Kyto University. It is a powerful and complex system. Wnn4 is free and Wnn6 is not. Wnn uses grammatical hints and probability of word association, so in principle Wnn can be cleverer than other methods.

Canna, made by NEC, supports only Japanese. It is a simple and powerful system. Canna uses only grammar, but its grammar and dictionary are quite sophisticated. So for standard modern Japanese, Canna seems cleverer than Wnn4. In addition, the UNIX version of Canna is free (now there is a Microsoft Windows version).

SJ3, by Sony, supports only Japanese.

Egg consists of following parts:

1. Input character Translation System (ITS) layer. It translates ASCII inputs to Kana/PinYin/Hangul characters.
2. Kana/PinYin/Hangul to Kanji transfer layer. The interface layer to network Kana-Kanji server (Wnn and Sj3).

These input methods are modal. They have a raw (alphabet) mode, a phonetic input mode, and Kana-Kanji transfer mode. However there are mode-less input methods for Egg and Canna. '`boiled-egg`' is a mode-less input method running on Egg. For Canna, '`canna.el`' has a tiny boiled-egg-like command, `(canna-boil)`, and there are some boiled-egg-like utilities.

Much of this information was provided by MORIOKA Tomohiko.

Q1.3.6: How do I portably code for MULE/XEmacs?

MULE has evolved rapidly over the last few years, and the original third party patch (for GNU Emacs 19), GNU Emacs 20+, and XEmacs 20+ have quite different implementations. The APIs also vary although recent versions of XEmacs have tended to converge to the GNU Emacs standard.

MULE implementations are going to continue to evolve. Both GNU Emacs and XEmacs are working hard on Unicode support, which will involve new APIs and probably variations on old ones. For XEmacs 22, the old ISO 2022-based system for recognizing encodings will be replaced by a much more flexible system, which should improve accuracy of automatic coding detections, but will also involve new APIs.

MORIOKA Tomohiko writes:

> The application implementor must write separate code for these mule variants. [Please don't hesitate to report these variants to us; they are not, strictly speaking, bugs, but they give third-party developers the same kind of creepy-crawly feeling. We'll do what we can. – Ed.]
>
> MULE and the next version of Emacs are similar but the symbols are very different—requiring separate code as well.
>
> Namely we must support 3 kinds of mule variants and 4 or 5 or 6 kinds of emacs variants... (;_;) I'm shocked, so I wrote a wrapper package called `emu` to provide a common interface. [There is an XEmacs package of APEL which provides much more comprehensive coverage. Be careful, however; APEL has problems of its own. – Ed.]
>
> I have the following suggestions about dealing with mule variants:
>
> - `(featurep 'mule) t` on all mule variants
> - `(boundp 'MULE)` is `t` on only MULE. Maybe the next version of Emacs will not have this symbol.
> - MULE has a variable `mule-version`. Perhaps the next version of Emacs will have this variable as well.
>
> Following is a sample to distinguish mule variants:
>
> ```
> (if (featurep 'mule)
>     (cond ((boundp 'MULE)
>            ;; for original Mule
>            )
>           ((string-match "XEmacs" emacs-version)
>            ;; for XEmacs with Mule
>            )
>           (t
>            ;; for next version of Emacs
>            ))
>   ;; for old emacs variants
>   )
> ```

Q1.3.7: How about Cyrillic Modes?

Ilya Zakharevich writes:

> There is a cyrillic mode in the file '`mysetup.zip`' in `ftp://ftp.math.ohio-state.edu/pub/users/ilya/emacs/`. This is a modification to Valery Alexeev's '`russian.el`' which can be obtained from

`http://www.math.uga.edu/~valery/russian.el`.

Dima Barsky writes:

> There is another cyrillic mode for both GNU Emacs and XEmacs by Dmitrii (Mitya) Manin at

> `http://kulichki-lat.rambler.ru/centrolit/manin/cyr.el`.

Rebecca Ore writes:

> The fullest resource I found on Russian language use (in and out of XEmacs) is `http://www.ibiblio.org/sergei/Software/Software.html`

Q1.3.8: Does XEmacs support Unicode?

Partially, as an external encoding for files, processes, and terminals. It does not yet support Unicode fonts [Q1.3.9], page 10

To get Unicode support, you need a Mule-enabled XEmacs. Install Mule-UCS from packages in the usual way. Put

```
(require 'un-define)
(set-coding-priority-list '(utf-8))
(set-coding-category-system 'utf-8 'utf-8)
```

in your init file to enable the UTF-8 coding system. You may wish to view the documentation of `set-coding-priority-list` if you find that files that are not UTF-8 are being mis-recognized as UTF-8.

Install standard national fonts (not Unicode fonts) for all character sets you use. See [Q1.3.9], page 10.

Mule-UCS also supports 16-bit forms of Unicode (UTF-16). It does not support 31-bit forms of Unicode (UTF-32 or UCS-4).

Q1.3.9: How does XEmacs display Unicode?

Mule doesn't have a Unicode charset internally, so there's nothing to bind a Unicode registry to. It would not be straightforward to create, either, because Unicode is not ISO 2022-compatible. You'd have to translate it to multiple 96x96 pages.

This means that Mule-UCS uses ordinary national fonts for display. This is not really a problem, except for those languages that use the Unified Han characters. The problem here is that Mule-UCS maps from Unicode code points to national character sets in a deterministic way. By default, this means that Japanese fonts are tried first, then Chinese, then Korean. To change the priority ordering, use the command 'un-define-change-charset-order'.

It also means you can't use Unicode fonts directly, at least not without extreme hackery. You can run -nw with (set-terminal-coding-system 'utf-8) if you really want a Unicode font for some reason.

Real Unicode support will be introduced in XEmacs 22.0.

1.4: Getting Started, Backing up & Recovery

Q1.4.1: What is an ` init.el ' or ` .emacs' and is there a sample one?

The '`init.el`' or '`.emacs`' file is used to customize XEmacs to your tastes. Starting in 21.4, the preferred location for the init file is '`~/.xemacs/init.el`'; in previous versions, it was '`~/.emacs`'. 21.4 still accepts the old location, but the first time you run it, it will ask to migrate your file to the new location. If you answer yes, the file will be moved, and a "compatibility" '`.emacs`' file will be placed in the old location so that you can still run older versions of XEmacs, and versions of GNU Emacs, which expect the old location. The '`.emacs`' file present is just a stub that loads the real file in '`~/.xemacs/init.el`'.

No two init files are alike, nor are they expected to be alike, but that's the point. The XEmacs distribution contains an excellent starter example in the '`etc/`' directory called '`sample.init.el`' (starting in 21.4) or '`sample.emacs`' in older versions. Copy this file from there to '`~/.xemacs/init.el`' (starting in 21.4) or '`~/.emacs`' in older versions, where '`~`' means your home directory, of course. Then edit it to suit.

You may bring the '`sample.init.el`' or '`sample.emacs`' file into an XEmacs buffer from the menubar. (The menu entry for it is always under the '`Help`' menu, but its location under that has changed in various versions. Recently, look under the '`Samples`' submenu.) To determine the location of the '`etc/`' directory type the command C-h v data-directory ⟨RET⟩.

Q1.4.2: Can I use the same ` init.el '/` .emacs' with the other Emacs?

Yes. The sample '`init.el`'/'`.emacs`' included in the XEmacs distribution will show you how to handle different versions and flavors of Emacs.

Q1.4.3: Any good tutorials around?

There's the XEmacs tutorial available from the Help Menu under '`Basics->Tutorials`', or by typing C-h t. To check whether it's available in a non-english language, type C-u C-h t TAB, type the first letters of your preferred language, then type ⟨RET⟩.

Q1.4.4: May I see an example of a useful XEmacs Lisp function?

The following function does a little bit of everything useful. It does something with the prefix argument, it examines the text around the cursor, and it's interactive so it may be bound to a key. It inserts copies of the current word the cursor is sitting on at the cursor. If you give it a prefix argument: C-u 3 M-x double-word then it will insert 3 copies.

```
(defun double-word (count)
  "Insert a copy of the current word underneath the cursor"
  (interactive "*p")
  (let (here there string)
    (save-excursion
      (forward-word -1)
      (setq here (point))
      (forward-word 1)
      (setq there (point))
      (setq string (buffer-substring here there)))
    (while (>= count 1)
      (insert string)
      (decf count))))
```

The best way to see what is going on here is to let XEmacs tell you. Put the code into an XEmacs buffer, and do a C-h f with the cursor sitting just to the right of the function you want explained. Eg. move the cursor to the SPACE between `interactive` and '`"*p"`' and hit C-h f to see what the function `interactive` does. Doing this will tell you that the `*` requires a writable buffer, and `p` converts the prefix argument to a number, and `interactive` allows you to execute the command with M-x.

Q1.4.5: And how do I bind it to a key?

To bind to a key do:

```
(global-set-key "\C-cd" 'double-word)
```

Or interactively, M-x global-set-key and follow the prompts.

Q1.4.6: What's the difference between a macro and a function?

Quoting from the Lisp Reference (a.k.a Lispref) Manual:

Macros enable you to define new control constructs and other language features. A macro is defined much like a function, but instead of telling how to compute a value, it tells how to compute another Lisp expression which will in turn compute the value. We call this expression the expansion of the macro.

Macros can do this because they operate on the unevaluated expressions for the arguments, not on the argument values as functions do. They can therefore construct an expansion containing these argument expressions or parts of them.

Do not confuse the two terms with keyboard macros, which are another matter, entirely. A keyboard macro is a key bound to several other keys. Refer to manual for details.

2 Installation and Trouble Shooting

This is part 2 of the XEmacs Frequently Asked Questions list. This section is devoted to Installation, Maintenance and Trouble Shooting.

2.0: Installation

Q2.0.1: Running XEmacs without installing

How can I just try XEmacs without installing it?

XEmacs will run in place without requiring installation and copying of the Lisp directories, and without having to specify a special build-time flag. It's the copying of the Lisp directories that requires so much space. XEmacs is largely written in Lisp.

A good method is to make a shell alias for xemacs:

```
alias xemacs=/i/xemacs-20.2/src/xemacs
```

(You will obviously use whatever directory you downloaded the source tree to instead of '`/i/xemacs-20.2`').

This will let you run XEmacs without massive copying.

Q2.0.2: XEmacs is too big

The space required by the installation directories can be reduced dramatically if desired. Gzip all the .el files. Remove all the packages you'll never want to use. Remove the TexInfo manuals. Remove the Info (and use just hardcopy versions of the manual). Remove most of the stuff in etc. Remove or gzip all the source code. Gzip or remove the C source code. Configure it so that copies are not made of the support lisp.

These are all Emacs Lisp source code and bytecompiled object code. You may safely gzip everything named *.el here. You may remove any package you don't use. *Nothing bad will happen if you delete a package that you do not use.* You must be sure you do not use it though, so be conservative at first.

Any package with the possible exceptions of xemacs-base, and EFS are candidates for removal. Ask yourself, *Do I ever want to use this package?* If the answer is no, then it is a candidate for removal.

First, gzip all the .el files. Then go about package by package and start gzipping the .elc files. Then run XEmacs and do whatever it is you normally do. If nothing bad happens, then remove the package. You can remove a package via the PUI interface (`M-x pui-list-packages`, then press `d` to mark the packages you wish to delete, and then `x` to delete them.

Another method is to do `M-x package-get-delete-package`.

Q2.0.3: Compiling XEmacs with Netaudio.

What is the best way to compile XEmacs with the netaudio system, since I have got the netaudio system compiled but installed at a weird place, I am not root. Also in the READMEs it does not say anything about compiling with the audioserver?

You should only need to add some stuff to the configure command line. To tell it to compile in netaudio support: '`--with-sound=both`', or '`--with-sound=nas`' if you don't want native sound support for some reason.) To tell it where to find the netaudio includes and libraries:

```
--site-libraries=WHATEVER
--site-includes=WHATEVER
```

Then (fingers crossed) it should compile and it will use netaudio if you have a server running corresponding to the X server. The netaudio server has to be there when XEmacs starts. If the netaudio server goes away and another is run, XEmacs should cope (fingers crossed, error handling in netaudio isn't perfect).

BTW, netaudio has been renamed as it has a name clash with something else, so if you see references to NAS or Network Audio System, it's the same thing. It also might be found at `ftp://ftp.x.org/contrib/audio/nas/`.

Q2.0.4: Problems with Linux and ncurses.

On Linux 1.3.98 with termcap 2.0.8 and the ncurses that came with libc 5.2.18, XEmacs 20.0b20 is unable to open a tty device:

```
src/xemacs -nw -q
Initialization error:
```

```
Terminal type ‘xterm’ undefined (or can’t access database?)
```

Ben Wing writes:

> Your ncurses configuration is messed up. Your /usr/lib/terminfo is a bad pointer, perhaps to a CD-ROM that is not inserted.

Q2.0.5: Do I need X11 to run XEmacs?

No. The name XEmacs is unfortunate in the sense that it is **not** an X Window System-only version of Emacs. XEmacs has full color support on a color-capable character terminal.

Q2.0.6: I'm having strange crashes. What do I do?

There have been a variety of reports of crashes due to compilers with buggy optimizers. Please see the ‘`PROBLEMS`’ file that comes with XEmacs to read what it says about your platform.

If you compiled XEmacs using ‘`--use-union-type`’ (or the option ‘`USE_UNION_TYPE`’ in ‘`config.inc`’ under Windows), try recompiling again without it. The union type has been known to trigger compiler errors in a number of cases.

Q2.0.7: Libraries in non-standard locations

I have x-faces, jpeg, xpm etc. all in different places. I’ve tried space-separated, comma-separated, several –site-libraries, all to no avail.

```
--site-libraries=’/path/one /path/two /path/etc’
```

Q2.0.8: can't resolve symbol _h_errno

You are using the Linux/ELF distribution of XEmacs 19.14, and your ELF libraries are out of date. You have the following options:

1. Upgrade your libc to at least 5.2.16 (better is 5.2.18, 5.3.12, or 5.4.10).
2. Patch the XEmacs binary by replacing all occurrences of ‘`_h_errno^@`’ with ‘`h_errno^@^@`’. Any version of Emacs will suffice. If you don’t understand how to do this, don’t do it.
3. Rebuild XEmacs yourself—any working ELF version of libc should be O.K.

Hrvoje Niksic writes:

> Why not use a Perl one-liner for No. 2?
>
> ```
> perl -pi -e ’s/_h_errno\0/h_errno\0\0/g’ \
> /usr/local/bin/xemacs-19.14
> ```
>
> NB: You *must* patch ‘`/usr/local/bin/xemacs-19.14`’, and not ‘`xemacs`’ because ‘`xemacs`’ is a link to ‘`xemacs-19.14`’; the Perl ‘`-i`’ option will cause unwanted side-effects if applied to a symbolic link.

SL Baur writes:

> If you build against a recent libc-5.4 (late enough to have caused problems earlier in the beta cycle) and then run with an earlier version of libc, you get a

```
$ xemacs
xemacs: can't resolve symbol '__malloc_hook'
zsh: 7942 segmentation fault (core dumped)  xemacs
```

(Example binary compiled against libc-5.4.23 and run with libc-5.4.16).

The solution is to upgrade to at least libc-5.4.23. Sigh. Drat.

Q2.0.9: Where do I find external libraries?

All external libraries used by XEmacs can be found at the XEmacs FTP site ftp://ftp.xemacs.org/pub/xemacs/aux/. [These tarballs and this FAQ are wa-a-ay out of date. Sorry, I'm not currently network-capable, and I will probably forgot to update this before submitting the patch. – Ed.]

The canonical locations (at the time of this writing) are as follows:

JPEG	ftp://ftp.uu.net/graphics/jpeg/. Version 6a is current.
XPM	ftp://ftp.x.org/contrib/libraries/. Version 3.4j is current. Older versions of this package are known to cause XEmacs crashes.
TIFF	ftp://ftp.sgi.com/graphics/tiff/. v3.4 is current. The latest beta is v3.4b035. There is a HOWTO here.
PNG	ftp://ftp.uu.net/graphics/png/. 0.89c is current. XEmacs requires a fairly recent version to avoid using temporary files. ftp://swrinde.nde.swri.edu/pub/png/src/
Compface	ftp://ftp.cs.indiana.edu/pub/faces/compface/. This library has been frozen for about 6 years, and is distributed without version numbers. *It should be compiled with the same options that X11 was compiled with on your system.* The version of this library at XEmacs.org includes the '`xbm2xface.pl`' script, written by stig@hackvan.com, which may be useful when generating your own xface.
NAS	ftp://ftp.x.org/contrib/audio/nas/. Version 1.2p5 is current. There is a FAQ here.

Q2.0.10: After I run configure I find a core dump, is something wrong?

Not necessarily. If you have GNU sed 3.0 you should downgrade it to 2.05. From the '`README`' at prep.ai.mit.edu:

> sed 3.0 has been withdrawn from distribution. It has major revisions, which mostly seem to be improvements; but it turns out to have bugs too which cause trouble in some common cases.
>
> Tom Lord won't be able to work fixing the bugs until May. So in the mean time, we've decided to withdraw sed 3.0 from distribution and make version 2.05 once again the recommended version.

It has also been observed that the vfork test on Solaris will leave a core dump.

Q2.0.11: XEmacs doesn't resolve hostnames.

This is the result of a long-standing problem with SunOS and the fact that stock SunOS systems do not ship with DNS resolver code in libc.

Christopher Davis writes:

> That's correct [The SunOS 4.1.3 precompiled binaries don't do name lookup]. Since Sun figured that everyone used NIS to do name lookups (that DNS thing was apparently only a passing fad, right?), the stock SunOS 4.x systems don't have DNS-based name lookups in libc.
>
> This is also why Netscape ships two binaries for SunOS 4.1.x.
>
> The best solution is to compile it yourself; the configure script will check to see if you've put DNS in the shared libc and will then proceed to link against the DNS resolver library code.

Q2.0.12: Why can't I strip XEmacs?

Richard Cognot writes:

> Because of the way XEmacs (and every other Emacsen, AFAIK) is built. The link gives you a bare-boned emacs (called temacs). temacs is then run, preloading some of the lisp files. The result is then dumped into a new executable, named xemacs, which will contain all of the preloaded lisp functions and data.
>
> Now, during the dump itself, the executable (code+data+symbols) is written on disk using a special unexec() function. This function is obviously heavily system dependent. And on some systems, it leads to an executable which, although valid, cannot be stripped without damage. If memory serves, this is especially the case for AIX binaries. On other architectures it might work OK.
>
> The Right Way to strip the emacs binary is to strip temacs prior to dumping xemacs. This will always work, although you can do that only if you install from sources (as temacs is '`not`' part of the binary kits).

Nat Makarevitch writes:

> Here is the trick:
>
> 1. [./configure; make]
> 2. rm src/xemacs
> 3. strip src/temacs
> 4. make
> 5. cp src/xemacs /usr/local/bin/xemacs
> 6. cp lib-src/DOC-19.16-XEmacs \
>   /usr/local/lib/xemacs-19.16/i586-unknown-linuxaout

Q2.0.13: I don't need no steenkin' packages. Do I?

Strictly speaking, no. XEmacs will build and install just fine without any packages installed. However, only the most basic editing functions will be available with no packages installed, so installing packages is an essential part of making your installed XEmacs _useful_.

Q2.0.14: How do I figure out which packages to install?

Many people really liked the old way that packages were bundled and do not want to mess with packages at all. You can grab all the packages at once like you used to with old XEmacs versions. Download the file

‘`xemacs-sumo.tar.gz`’

For an XEmacs compiled with Mule you also need

‘`xemacs-mule-sumo.tar.gz`’

from the ‘`packages`’ directory on your XEmacs mirror archive. N.B. They are called ’Sumo Tarballs’ for good reason. They are currently about 15MB and 2.3MB (gzipped) respectively.

Install them by

```
cd $prefix/lib/xemacs ; gunzip -c <tarballname> | tar xf -
```

See README.packages for more detailed installation instructions.

As the Sumo tarballs are not regenerated as often as the individual packages, it is recommended that you use the automatic package tools afterwards to pick up any recent updates.

Q2.0.15: EFS fails with "500 AUTH not understood" (NEW)

A typical error: FTP Error: USER request failed; 500 AUTH not understood.

Thanks to giacomo boffi `giacomo.boffi@polimi.it` who recommends on comp.emacs.xemacs:

tell your ftp client to not attempt AUTH authentication (or do not use FTP servers that don’t understand AUTH)

and notes that you need to add an element (often "-u") to ‘efs-ftp-program-args’. Use M-x customize-variable, and verify the needed flag with ‘man ftp’ or other local documentation.

Q2.0.16: Cygwin XEmacs won't start: cygXpm-noX4.dll was not found (NEW)

The Cygwin binary distributed with the netinstaller uses an external DLL to handle XPM images (such as toolbar buttons). You may get an error like

This application has failed to start because cygXpm-noX4.dll was not found. Re-installing the application may fix this problem.

Andy Piper <andy@xemacs.org> sez:

cygXpm-noX4 is part of the cygwin distribution under libraries or graphics, but is not installed by default. You need to run the cygwin setup again and select this package.

Ie, reinstalling XEmacs won’t help because it is not part of the XEmacs distribution.

2.1: Trouble Shooting

Q2.1.1: Help! XEmacs just crashed on me!

First of all, don't panic. Whenever XEmacs crashes, it tries extremely hard to auto-save all of your files before dying. (The main time that this will not happen is if the machine physically lost power or if you killed the XEmacs process using `kill -9`). The next time you try to edit those files, you will be informed that a more recent auto-save file exists. You can use M-x recover-file to retrieve the auto-saved version of the file.

You can use the command M-x recover-session after a crash to pick up where you left off.

Now, XEmacs is not perfect, and there may occasionally be times, or particular sequences of actions, that cause it to crash. If you can come up with a reproducible way of doing this (or even if you have a pretty good memory of exactly what you were doing at the time), the maintainers would be very interested in knowing about it. The best way to report a bug is using M-x report-emacs-bug (or by selecting '`Send Bug Report...`' from the Help menu). If that won't work (e.g. you can't get XEmacs working at all), send ordinary mail to `crashes@xemacs.org`. *MAKE SURE* to include the output from the crash, especially including the Lisp backtrace, as well as the XEmacs configuration from M-x describe-installation (or equivalently, the file '`Installation`' in the top of the build tree). Please note that the '`crashes`' address is exclusively for crash reports. The best way to report bugs in general is through the M-x report-emacs-bug interface just mentioned, or if necessary by emailing `xemacs-beta@xemacs.org`. Note that the developers do *not* usually follow '`comp.emacs.xemacs`' on a regular basis; thus, this is better for general questions about XEmacs than bug reports.

If at all possible, include a C stack backtrace of the core dump that was produced. This shows where exactly things went wrong, and makes it much easier to diagnose problems. To do this under Unix, you need to locate the core file (it's called '`core`', and is usually sitting in the directory that you started XEmacs from, or your home directory if that other directory was not writable). Then, go to that directory and execute a command like:

```
gdb `which xemacs` core
```

and then issue the command '`where`' to get the stack backtrace. You might have to use `dbx` or some similar debugger in place of `gdb`. If you don't have any such debugger available, complain to your system administrator.

It's possible that a core file didn't get produced, in which case you're out of luck. Go complain to your system administrator and tell him not to disable core files by default. Also see [Q2.1.15], page 23, for tips and techniques for dealing with a debugger.

If you're under Microsoft Windows, you're out of luck unless you happen to have a debugging aid installed on your system, for example Visual C++. In this case, the crash will result in a message giving you the option to enter a debugger (for example, by pressing '`Cancel`'). Do this and locate the stack-trace window. (If your XEmacs was built without debugging information, the stack trace may not be very useful.)

When making a problem report make sure that:

1. Report all of the information output by XEmacs during the crash.

2. You mention what O/S & Hardware you are running XEmacs on.
3. What version of XEmacs you are running.
4. What build options you are using.
5. If the problem is related to graphics and you are running Unix, we will also need to know what version of the X Window System you are running, and what window manager you are using.
6. If the problem happened on a TTY, please include the terminal type.

Much of the information above is automatically generated by M-x report-emacs-bug . Even more, and often useful, information can be generated by redirecting the output of `make` and `make check` to a file (‘`beta.err`’ is the default used by `build-report`), and executing M-x build-report .

Q2.1.2: Cryptic Minibuffer messages.

When I try to use some particular option of some particular package, I get a cryptic error in the minibuffer.

If you can’t figure out what’s going on, select Options/General Options/Debug on Error from the Menubar and then try and make the error happen again. This will give you a backtrace that may be enlightening. If not, try reading through this FAQ; if that fails, you could try posting to comp.emacs.xemacs (making sure to include the backtrace) and someone may be able to help. If you can identify which Emacs lisp source file the error is coming from you can get a more detailed stack backtrace by doing the following:

1. Visit the .el file in an XEmacs buffer.
2. Issue the command M-x eval-current-buffer .
3. Reproduce the error.

Depending on the version of XEmacs, you may either select View->Show Message Log (recent versions), Edit->Show Messages (some earlier versions) or Help->Recent Keystrokes/Messages (other earlier versions) from the menubar to see the most recent messages. This command is bound to C-h l by default.

Q2.1.3: Translation Table Syntax messages at Startup

I get tons of translation table syntax error messages during startup. How do I get rid of them?

There are two causes of this problem. The first usually only strikes people using the prebuilt binaries. The culprit in both cases is the file ‘`XKeysymDB`’.

- The binary cannot find the ‘`XKeysymDB`’ file. The location is hardcoded at compile time so if the system the binary was built on puts it a different place than your system does, you have problems. To fix, set the environment variable `XKEYSYMDB` to the location of the ‘`XKeysymDB`’ file on your system or to the location of the one included with XEmacs which should be at
‘`<xemacs_root_directory>/lib/xemacs-19.16/etc/XKeysymDB`’.
- The binary is finding the XKeysymDB but it is out-of-date on your system and does not contain the necessary lines. Either ask your system administrator to replace it with

the one which comes with XEmacs (which is the stock R6 version and is backwards compatible) or set your `XKEYSYMDB` variable to the location of XEmacs's described above.

Q2.1.4: Startup warnings about deducing proper fonts?

How can I avoid the startup warnings about deducing proper fonts?

This is highly dependent on your installation, but try with the following font as your base font for XEmacs and see what it does:

-adobe-courier-medium-r-*-*-*-120-*-*-*-*-iso8859-1

More precisely, do the following in your resource file:

Emacs.default.attributeFont: \
-adobe-courier-medium-r-*-*-*-120-*-*-*-*-iso8859-1

If you just don't want to see the '`*Warnings*`' buffer at startup time, you can set this:

```
(setq display-warning-minimum-level 'error)
```

The buffer still exists; it just isn't in your face.

Q2.1.5: XEmacs cannot connect to my X Terminal!

Help! I can not get XEmacs to display on my Envizex X-terminal!

Try setting the `DISPLAY` variable using the numeric IP address of the host you are running XEmacs from.

Q2.1.6: XEmacs just locked up my Linux X server!

There have been several reports of the X server locking up under Linux. In all reported cases removing speedo and scaled fonts from the font path corrected the problem. This can be done with the command `xset`.

It is possible that using a font server may also solve the problem.

Q2.1.7: HP Alt key as Meta.

How can I make XEmacs recognize the Alt key of my HP workstation as a Meta key?

Put the following line into a file and load it with xmodmap(1) before starting XEmacs:

```
remove Mod1 = Mode_switch
```

Q2.1.8: got (wrong-type-argument color-instance-p nil)

Natalie Kershaw writes:

> I am trying to run xemacs 19.13 under X11R4. Whenever I move the mouse I get the following error. Has anyone seen anything like this? This doesn't occur on X11R5.
>
> ```
> Signalling:
> (error "got (wrong-type-argument color-instance-p nil)
> and I don't know why!")
> ```

dinos writes:

> I think this is due to undefined resources; You need to define color backgrounds and foregrounds into your '`.../app-defaults/Emacs`' like:
>
> ```
> *Foreground:    Black   ;everything will be of black on grey95,
> *Background:    Grey95  ;unless otherwise specified.
> *cursorColor:   Red3    ;red3 cursor with grey95 border.
> *pointerColor:  Red3    ;red3 pointer with grey95 border.
> ```

Natalie Kershaw adds:

> What fixed the problem was adding some more colors to the X color database (copying the X11R5 colors over), and also defining the following resources:
>
> ```
> xemacs*cursorColor:    black
> xemacs*pointerColor:   black
> ```
>
> With the new colors installed the problem still occurs if the above resources are not defined.
>
> If the new colors are not present then an additional error occurs on XEmacs startup, which says '`Color Red3`' not defined.

Q2.1.9: XEmacs causes my OpenWindows 3.0 server to crash.

The OpenWindows 3.0 server is incredibly buggy. Your best bet is to replace it with one from the generic MIT X11 release. You might also try disabling parts of your '`init.el`'/'`.emacs`', like those that enable background pixmaps.

Q2.1.10: Warnings from incorrect key modifiers.

The following information comes from the '`PROBLEMS`' file that comes with XEmacs.

If you're having troubles with HP/UX it is because HP/UX defines the modifiers wrong in X. Here is a shell script to fix the problem; be sure that it is run after VUE configures the X server.

```
#! /bin/sh
xmodmap 2> /dev/null - << EOF
keysym Alt_L = Meta_L
keysym Alt_R = Meta_R
EOF

xmodmap - << EOF
clear mod1
keysym Mode_switch = NoSymbol
add mod1 = Meta_L
keysym Meta_R = Mode_switch
add mod2 = Mode_switch
EOF
```

Q2.1.11: `Can't instantiate image error... ' in toolbar

Dr. Ram Samudrala writes:

I just installed the XEmacs (20.4-2) RPMS that I downloaded from http://www.xemacs.org/. Everything works fine, except that when I place my mouse over the toolbar, it beeps and gives me this message:

```
Can't instantiate image (probably cached):
[xbm :mask-file "/usr/include/X11/bitmaps/leftptrmsk :mask-data
(16 16 <strange control characters> ...
```

Kyle Jones writes:

> This is problem specific to some Chips and Technologies video chips, when running XFree86. Putting
>
> `Option "sw_cursor"`
>
> in '`XF86Config`' gets rid of the problem.

Q2.1.12: Problems with Regular Expressions on DEC OSF1.

I have xemacs 19.13 running on an alpha running OSF1 V3.2 148 and ispell would not run because it claimed the version number was incorrect although it was indeed OK. I traced the problem to the regular expression handler.

Douglas Kosovic writes:

> Actually it's a DEC cc optimization bug that screws up the regexp handling in XEmacs.
>
> Rebuilding using the '`-migrate`' switch for DEC cc (which uses a different sort of optimization) works fine.

See '`xemacs-19_13-dunix-3_2c.patch`' at the following URL on how to build with the '`-migrate`' flag:

http://www-digital.cern.ch/carney/emacs/emacs.html

NOTE: There have been a variety of other problems reported that are fixed in this fashion.

Q2.1.13: HP/UX 10.10 and create_process failure.

Dave Carrigan writes:

> With XEmacs 19.13 and HP/UX 10.10, anything that relies on the `create_process` function fails. This breaks a lot of things (shell-mode, compile, ange-ftp, to name a few).

Phil Johnson writes:

> This is a problem specific to HP-UX 10.10. It only occurs when XEmacs is compiled for shared libraries (the default), so you can work around it by compiling a statically-linked binary (run configure with '`--dynamic=no`').
>
> I'm not sure whether the problem is with a particular shared library or if it's a kernel problem which crept into 10.10.

Richard Cognot writes:

> I had a few problems with 10.10. Apparently, some of them were solved by forcing a static link of libc (manually).

Q2.1.14: C-g doesn't work for me. Is it broken?

Ben Wing writes:

> C-g does work for most people in most circumstances. If it doesn't, there are only two explanations:
>
> 1. The code is wrapped with a binding of `inhibit-quit` to `t`. Ctrl-Shift-G should still work, I think.
> 2. SIGIO is broken on your system, but BROKEN_SIGIO isn't defined.
>
> To test #2, try executing `(while t)` from the '`*scratch*`' buffer. If C-g doesn't interrupt, then you're seeing #2.

Morten Welinder writes:

> On some (but *not* all) machines a hung XEmacs can be revived by `kill -FPE <pid>`. This is a hack, of course, not a solution. This technique works on a Sun4 running 4.1.3_U1. To see if it works for you, start another XEmacs and test with that first. If you get a core dump the method doesn't work and if you get '`Arithmetic error`' then it does.

Q2.1.15: How to debug an XEmacs problem with a debugger

If XEmacs does crash on you, one of the most productive things you can do to help get the bug fixed is to poke around a bit with the debugger. Here are some hints:

- First of all, if the crash is at all reproducible, consider very strongly recompiling your XEmacs with debugging symbols and with no optimization (e.g. with GCC use the compiler flags '`-g -O0`' – that's an "oh" followed by a zero), and with the configure options '`--debug=yes`' and '`--error-checking=all`'. This will make your XEmacs run somewhat slower, but you are a lot more likely to catch the problem earlier (closer to its source). It makes it a lot easier to determine what's going on with a debugger.
- If it's not a true crash (*i.e.*, XEmacs is hung, or a zombie process), or it's inconvenient to run XEmacs again because XEmacs is already running or is running in batch mode as part of a bunch of scripts, you may be able to attach to the existing process with your debugger. Most debuggers let you do this by substituting the process ID for the core file when you invoke the debugger from the command line, or by using the `attach` command or something similar.
- If you're able to run XEmacs under a debugger and reproduce the crash, here are some things you can do:
- If XEmacs is hitting an assertion failure, put a breakpoint on `assert_failed()`.
- If XEmacs is hitting some weird Lisp error that's causing it to crash (e.g. during startup), put a breakpoint on `signal_1()`—this is declared static in eval.c.
- If XEmacs is outputting lots of X errors, put a breakpoint on `x_error_handler()`; that will tell you which call is causing them.
- Internally, you will probably see lots of variables that hold objects of type `Lisp_Object`. These are references to Lisp objects. Printing them out with the debugger probably won't be too useful—you'll likely just see a number. To decode them, do this:

  ```
  call dp (OBJECT)
  ```

where OBJECT is whatever you want to decode (it can be a variable, a function call, etc.). This uses the Lisp printing routines to out a readable representation on the TTY from which the xemacs process was invoked.

- If you want to get a Lisp backtrace showing the Lisp call stack, do this:

  ```
  call db ()
  ```

- Using `dp` and `db` has two disadvantages - they can only be used with a running (including hung or zombie) xemacs process, and they do not display the internal C structure of a Lisp Object. Even if all you've got is a core dump, all is not lost.

  If you're using GDB, there are some macros in the file '`src/.gdbinit`' in the XEmacs source distribution that should make it easier for you to decode Lisp objects. This file is automatically read by gdb if gdb is run in the directory where xemacs was built, and contains these useful macros to inspect the state of xemacs:

  `pobj` Usage: pobj lisp_object
  Print the internal C representation of a lisp object.

  `xtype` Usage: xtype lisp_object
  Print the Lisp type of a lisp object.

  `lbt` Usage: lbt
  Print the current Lisp stack trace. Requires a running xemacs process. (It works by calling the db routine described above.)

  `ldp` Usage: ldp lisp_object
  Print a Lisp Object value using the Lisp printer. Requires a running xemacs process. (It works by calling the dp routine described above.)

  `run-temacs`
  Usage: run-temacs
  Run temacs interactively, like xemacs. Use this with debugging tools (like purify) that cannot deal with dumping, or when temacs builds successfully, but xemacs does not.

  `dump-temacs`
  Usage: dump-temacs
  Run the dumping part of the build procedure. Use when debugging temacs, not xemacs! Use this when temacs builds successfully, but xemacs does not.

  `check-xemacs`
  Usage: check-xemacs
  Run the test suite. Equivalent to 'make check'.

  `check-temacs`
  Usage: check-temacs
  Run the test suite on temacs. Equivalent to 'make check-temacs'. Use this with debugging tools (like purify) that cannot deal with dumping, or when temacs builds successfully, but xemacs does not.

  If you are using Sun's '`dbx`' debugger, there is an equivalent file '`src/.dbxrc`', which defines the same commands for dbx.

- If you're using a debugger to get a C stack backtrace and you're seeing stack traces with some of the innermost frames mangled, it may be due to dynamic linking. (This happens especially under Linux.) Consider reconfiguring with '`--dynamic=no`'. Also, sometimes (again under Linux), stack backtraces of core dumps will have the frame where the fatal signal occurred mangled; if you can obtain a stack trace while running the XEmacs process under a debugger, the stack trace should be clean.

  Curtiss suggests upgrading to ld.so version 1.8 if dynamic linking and debugging is a problem on Linux.
- If you're using a debugger to get a C stack backtrace and you're getting a completely mangled and bogus stack trace, it's probably due to one of the following:
  a. Your executable has been stripped. Bad news. Tell your sysadmin not to do this—it doesn't accomplish anything except to save a bit of disk space, and makes debugging much much harder.
  b. Your stack is getting trashed. Debugging this is hard; you have to do a binary-search type of narrowing down where the crash occurs, until you figure out exactly which line is causing the problem. Of course, this only works if the bug is highly reproducible. Also, in many cases if you run XEmacs from the debugger, the debugger can protect the stack somewhat. However, if the stack is being smashed, it is typically the case that there is a wild pointer somewhere in the program, often quite far from where the crash occurs.
  c. If your stack trace has exactly one frame in it, with address 0x0, this could simply mean that XEmacs attempted to execute code at that address, e.g. through jumping to a null function pointer. Unfortunately, under those circumstances, GDB under Linux doesn't know how to get a stack trace. (Yes, this is the fourth Linux-related problem I've mentioned. I have no idea why GDB under Linux is so bogus. Complain to the GDB authors, or to comp.os.linux.development.system.) Again, you'll have to use the narrowing-down process described above.
  d. You will get a Lisp backtrace output when XEmacs crashes, so you'll have something useful.
- If you compile with the newer gcc variants gcc-2.8 or egcs, you will also need gdb 4.17 or above. Earlier releases of gdb can't handle the debug information generated by the newer compilers.
- In versions of XEmacs before 21.2.27, '`src/.gdbinit`' was named '`src/gdbinit`'. This had the disadvantage of not being sourced automatically by gdb, so you had to set that up yourself.
- If you are running Microsoft Windows, the the file '`nt/README`' for further information about debugging XEmacs.

Q2.1.16: XEmacs crashes in `strcat` on HP/UX 10

From the problems database (through the former address http://support.mayfield.hp.com/):

```
Problem Report: 5003302299
Status:         Open

System/Model:   9000/700
```

```
Product Name:   HPUX S800 10.0X
Product Vers:   9245XB.10.00

Description: strcat(3C) may read beyond
end of source string, can cause SIGSEGV


*** PROBLEM TEXT ***
strcat(3C) may read beyond the source string onto an unmapped page,
causing a segmentation violation.
```

Q2.1.17: `Marker does not point anywhere'

As with other errors, set `debug-on-error` to `t` to get the backtrace when the error occurs. Specifically, two problems have been reported (and fixed).

1. A problem with line-number-mode in XEmacs 19.14 affected a large number of other packages. If you see this error message, turn off line-number-mode.
2. A problem with some early versions of Gnus 5.4 caused this error. Upgrade your Gnus.

Q2.1.18: XEmacs is outputting lots of X errors.

If this is happening, we would very much like to know what's causing them. To find this out, see [Q2.1.15], page 23. Try to get both a C and Lisp backtrace, and send them to `xemacs-beta@xemacs.org`.

Q2.1.19: XEmacs does not follow the local timezone.

When using one of the prebuilt binaries many users have observed that XEmacs uses the timezone under which it was built, but not the timezone under which it is running. The solution is to add:

```
(set-time-zone-rule "MET")
```

to your '`init.el`'/'`.emacs`' or the '`site-start.el`' file if you can. Replace `MET` with your local timezone.

Q2.1.20: `Symbol's function definition is void: hkey-help-show. '

This is a problem with a partially loaded hyperbole. Try adding:

```
(require 'hmouse-drv)
```

where you load hyperbole and the problem should go away.

Q2.1.21: [This question intentionally left blank]

Q2.1.22: XEmacs seems to take a really long time to do some things

David Moore writes:

> Two things you can do:
>
> 1) C level:
>
> When you see it going mad like this, you might want to use gdb from an 'xterm' to attach to the running process and get a stack trace. To do this just run:
>
> ```
> gdb /path/to/xemacs/xemacs ####
> ```
>
> Where `####` is the process id of your xemacs, instead of specifying the core. When gdb attaches, the xemacs will stop [1] and you can type 'where' in gdb to get a stack trace as usual. To get things moving again, you can just type 'quit' in gdb. It'll tell you the program is running and ask if you want to quit anyways. Say 'y' and it'll quit and have your emacs continue from where it was at.
>
> 2) Lisp level:
>
> Turn on debug-on-quit early on. When you think things are going slow hit C-g and it may pop you in the debugger so you can see what routine is running. Press 'c' to get going again.
>
> debug-on-quit doesn't work if something's turned on inhibit-quit or in some other strange cases.

Q2.1.23: Movemail on Linux does not work for XEmacs 19.15 and later.

Movemail used to work fine in 19.14 but has stopped working in 19.15 and 20.x. I am using Linux.

SL Baur writes:

> Movemail on Linux used to default to using flock file locking. With 19.15 and later versions it now defaults to using `.lock` file locking. If this is not appropriate for your system, edit src/s/linux.h and uncomment the line that reads:
>
> ```
> #define MAIL_USE_FLOCK
> ```

Q2.1.24: XEmacs won't start without network.

If XEmacs starts when you're on the network, but fails when you're not on the network, you may be missing a "localhost" entry in your '`/etc/hosts`' file. The file should contain an entry like:

```
127.0.0.1          localhost
```

Add that line, and XEmacs will be happy.

Q2.1.25:: After upgrading, XEmacs won't do `foo' any more!

You have been used to doing 'foo', but now when you invoke it (or click the toolbar button or select the menu item), nothing (or an error) happens. The simplest explanation is that you are missing a package that is essential to you. You can either track it down and install it (there is a list of packages and brief descriptions of their contents in '`etc/PACKAGES`'), or install the 'Sumo Tarball' (see [Q2.0.14], page 17).

3 Customization and Options

This is part 3 of the XEmacs Frequently Asked Questions list. This section is devoted to Customization and screen settings.

3.0: Customization – Emacs Lisp and ‘init.el’/‘.emacs’

Q3.0.1: What version of Emacs am I running?

How can ‘`init.el`’/‘`.emacs`’ determine which of the family of Emacsen I am using?

To determine if you are currently running GNU Emacs 18, GNU Emacs 19, XEmacs 19, XEmacs 20, or Epoch, and use appropriate code, check out the example given in ‘`etc/sample.init.el`’ (‘`etc/sample.emacs`’ in XEmacs versions prior to 21.4). There are other nifty things in there as well!

For all new code, all you really need to do is:

```
(defvar running-xemacs (string-match "XEmacs\\|Lucid" emacs-version))
```

Q3.0.2: How can I evaluate Emacs-Lisp expressions?

I know I can evaluate Elisp expressions from `*scratch*` buffer with C-j after the expression. How do I do it from another buffer?

Press M-: (the default binding of `eval-expression`), and enter the expression to the minibuffer.

Q3.0.3: (setq tab-width 6) behaves oddly.

If you put `(setq tab-width 6)` in your ‘`init.el`’/‘`.emacs`’ file it does not work! Is there a reason for this? If you do it at the EVAL prompt it works fine!! How strange.

Use `setq-default` instead, since `tab-width` is all-buffer-local.

Q3.0.4: How can I add directories to the load-path ?

Here are two ways to do that, one that puts your directories at the front of the load-path, the other at the end:

```
;;; Add things at the beginning of the load-path, do not add
;;; duplicate directories:
(pushnew "bar" load-path :test 'equal)

(pushnew "foo" load-path :test 'equal)

;;; Add things at the end, unconditionally
(setq load-path (nconc load-path '("foo" "bar")))
```

keith (k.p.) hanlan writes:

> To add directories using Unix shell metacharacters use ‘`expand-file-name`’ like this:
>
> ```
> (push (expand-file-name "~keithh/.emacsdir") load-path)
> ```

Q3.0.5: How to check if a lisp function is defined?

Use the following elisp:

```
(fboundp 'foo)
```

It's almost always a mistake to test `emacs-version` or any similar variables.

Instead, use feature-tests, such as `featurep`, `boundp`, `fboundp`, or even simple behavioral tests, eg.:

```
(defvar foo-old-losing-code-p
  (condition-case nil (progn (losing-code t) nil)
    (wrong-number-of-arguments t)))
```

There is an incredible amount of broken code out there which could work much better more often in more places if it did the above instead of trying to divine its environment from the value of one variable.

Q3.0.6: Can I force the output of (face-list) to a buffer?

It would be good having it in a buffer, as the output of `(face-list)` is too wide to fit to a minibuffer.

Evaluate the expression in the '`*scratch*`' buffer with point after the rightmost paren and typing C-j .

If the minibuffer smallness is the only problem you encounter, you can simply press C-h l to get the former minibuffer contents in a buffer.

Q3.0.7: Font selections in don't get saved after Save Options.

John Mann writes:

> You have to go to Options->Frame Appearance and unselect '`Frame-Local Font Menu`'. If this option is selected, font changes are only applied to the *current* frame and do *not* get saved when you save options.

Also, set the following in your '`init.el`'/'`.emacs`':

```
(setq options-save-faces t)
```

Q3.0.8: How do I get a single minibuffer frame?

Vin Shelton writes:

```
(setq initial-frame-plist '(minibuffer nil))
(setq default-frame-plist '(minibuffer nil))
(setq default-minibuffer-frame
      (make-frame
       '(minibuffer only
                    width 86
                    height 1
                    menubar-visible-p nil
                    default-toolbar-visible-p nil
                    name "minibuffer"
                    top -2
```

```
                          left -2
                          has-modeline-p nil)))
(frame-notice-user-settings)
```

Please note: The single minibuffer frame may not be to everyone's taste, and there any number of other XEmacs options settings that may make it difficult or inconvenient to use.

Q3.0.9: What is Customize?

Starting with XEmacs 20.2 there is new system 'Customize' for customizing XEmacs options.

You can access `Customize` from the `Options` menu or invoking one of customize commands by typing eg. M-x customize, M-x customize-face , M-x customize-variable or M-x customize-apropos .

Starting with XEmacs 20.3 there is also new 'browser' mode for Customize. Try it out with M-x customize-browse

3.1: X Window System & Resources

Q3.1.1: Where is a list of X resources?

Search through the '`NEWS`' file for '`X Resources`'. A fairly comprehensive list is given after it.

In addition, an '`app-defaults`' file '`etc/Emacs.ad`' is supplied, listing the defaults. The file '`etc/sample.Xresources`' gives a different set of defaults that you might consider for installation in your '`~/.Xresources`' file. It is nearly the same as '`etc/Emacs.ad`', but a few entries are altered. Be careful about installing the contents of this file into your '`.Xresources`' (or legacy '`.Xdefaults`') file if you use GNU Emacs under X11 as well.

Q3.1.2: How can I detect a color display?

You can test the return value of the function `(device-class)`, as in:

```
(when (eq (device-class) 'color)
  (set-face-foreground  'font-lock-comment-face "Grey")
  (set-face-foreground  'font-lock-string-face  "Red")
  ....
  )
```

Q3.1.3: [This question intentionally left blank]

Q3.1.4: [This question intentionally left blank]

Q3.1.5: How can I get the icon to just say `XEmacs'?

I'd like the icon to just say '`XEmacs`', and not include the name of the current file in it.

Add the following line to your '`init.el`'/'`.emacs`':

```
(setq frame-icon-title-format "XEmacs")
```

Q3.1.6: How can I have the window title area display the full path?

I'd like to have the window title area display the full directory/name of the current buffer file and not just the name.

Add the following line to your '`init.el`'/'`.emacs`':

```
(setq frame-title-format "%S: %f")
```

A more sophisticated title might be:

```
(setq frame-title-format
      '("%S: " (buffer-file-name "%f"
                                 (dired-directory dired-directory "%b"))))
```

That is, use the file name, or the dired-directory, or the buffer name.

Q3.1.7: `xemacs -name junk' doesn't work?

When I run '`xterm -name junk`', I get an xterm whose class name according to xprop, is '`junk`'. This is the way it's supposed to work, I think. When I run '`xemacs -name junk`' the class name is not set to '`junk`'. It's still '`emacs`'. What does '`xemacs -name`' really do? The reason I ask is that my window manager (fvwm) will make a window sticky and I use XEmacs to read my mail. I want that XEmacs window to be sticky, without having to use the window manager's function to set the window sticky. What gives?

'`xemacs -name`' sets the application name for the program (that is, the thing which normally comes from '`argv[0]`'). Using '`-name`' is the same as making a copy of the executable with that new name. The `WM_CLASS` property on each frame is set to the frame-name, and the application-class. So, if you did '`xemacs -name FOO`' and then created a frame named BAR, you'd get an X window with WM_CLASS = `( "BAR", "Emacs")`. However, the resource hierarchy for this widget would be:

```
Name:    FOO   .shell             .container   .BAR
Class:   Emacs .TopLevelEmacsShell.EmacsManager.EmacsFrame
```

instead of the default

```
Name:    xemacs.shell             .container   .emacs
Class:   Emacs .TopLevelEmacsShell.EmacsManager.EmacsFrame
```

It is arguable that the first element of WM_CLASS should be set to the application-name instead of the frame-name, but I think that's less flexible, since it does not give you the ability to have multiple frames with different WM_CLASS properties. Another possibility would be for the default frame name to come from the application name instead of simply being '`emacs`'. However, at this point, making that change would be troublesome: it would mean that many users would have to make yet another change to their resource files (since the default frame name would suddenly change from '`emacs`' to '`xemacs`', or whatever the executable happened to be named), so we'd rather avoid it.

To make a frame with a particular name use:

```
(make-frame '((name . "the-name")))
```

Q3.1.8: `-iconic ' doesn't work.

When I start up XEmacs using '`-iconic`' it doesn't work right. Using '`-unmapped`' on the command line, and setting the `initiallyUnmapped` X Resource don't seem to help much either...

Ben Wing writes:

> Ugh, this stuff is such an incredible mess that I've about given up getting it to work. The principal problem is numerous window-manager bugs...

3.2: Textual Fonts & Colors

Q3.2.1: How can I set color options from ` init.el '/` .emacs'?

How can I set the most commonly used color options from my '`init.el`'/'`.emacs`' instead of from my '`.Xresources`'?

Like this:

```
(set-face-background 'default      "bisque") ; frame background
(set-face-foreground 'default      "black") ; normal text
(set-face-background 'zmacs-region "red") ; When selecting w/
                                          ; mouse
(set-face-foreground 'zmacs-region "yellow")
(set-face-font       'default      "*courier-bold-r*120-100-100*")
(set-face-background 'highlight    "blue") ; Ie when selecting
                                          ; buffers
(set-face-foreground 'highlight    "yellow")
(set-face-background 'modeline     "blue") ; Line at bottom
                                          ; of buffer
(set-face-foreground 'modeline     "white")
(set-face-font       'modeline     "*bold-r-normal*140-100-100*")
(set-face-background 'isearch      "yellow") ; When highlighting
                                          ; while searching
(set-face-foreground 'isearch      "red")
(setq x-pointer-foreground-color   "black") ; Adds to bg color,
                                          ; so keep black
(setq x-pointer-background-color   "blue") ; This is color
                                          ; you really
                                          ; want ptr/crsr
```

Q3.2.2: How do I set the text, menu and modeline fonts?

Note that you should use '`Emacs.`' and not '`Emacs*`' when setting face values.

In '`.Xresources`':

```
Emacs.default.attributeFont:  -*-*-medium-r-*-*-*-120-*-*-m-*-*-*
Emacs*menubar*font:           fixed
Emacs.modeline.attributeFont: fixed
```

This is confusing because ‘`default`’ and ‘`modeline`’ are face names, and can be found listed with all faces in the current mode by using M-x set-face-font (enter) ? . They use the face-specific resource ‘`attributeFont`’.

On the other hand, ‘`menubar`’ is a normal X thing that uses the resource ‘`font`’. With Motif it *may be* necessary to use ‘`fontList`’ *instead of* ‘`font`’. In *non-Motif* configurations with Mule it *is* necessary to use ‘`fontSet`’ instead of ‘`font`’. (Sorry, there just is no simple recipe here.)

Q3.2.3: How can I set the colors when highlighting a region?

How can I set the background/foreground colors when highlighting a region?

You can change the face `zmacs-region` either in your ‘`.Xresources`’:

```
Emacs.zmacs-region.attributeForeground: firebrick
Emacs.zmacs-region.attributeBackground: lightseagreen
```

or in your ‘`init.el`’/‘`.emacs`’:

```
(set-face-background 'zmacs-region "red")
(set-face-foreground 'zmacs-region "yellow")
```

Q3.2.4: How can I limit color map usage?

I’m using Netscape (or another color grabber like XEmacs); is there any way to limit the number of available colors in the color map?

Answer: No, but you can start Netscape before XEmacs, and it will use the closest available color if the colormap is full. You can also limit the number of colors Netscape uses, using the flags -mono, -ncols <#> or -install (for mono, limiting to <#> colors, or for using a private color map).

If you have the money, another solution would be to use a truecolor or direct color video.

Q3.2.5: My tty supports color, but XEmacs doesn't use them.

XEmacs tries to automatically determine whether your tty supports color, but sometimes guesses wrong. In that case, you can make XEmacs Do The Right Thing using this Lisp code:

```
(if (eq 'tty (device-type))
    (set-device-class nil 'color))
```

Q3.2.6: Can I have pixmap backgrounds in XEmacs?

Juan Villacis writes:

> There are several ways to do it. For example, you could specify a default pixmap image to use in your ‘`~/.Xresources`’, e.g.,
>
> ```
>      Emacs*EmacsFrame.default.attributeBackgroundPixmap: /path/to/image.xpm
> ```
>
> and then reload ~/.Xresources and restart XEmacs. Alternatively, since each face can have its own pixmap background, a better way would be to set a face’s pixmap within your XEmacs init file, e.g.,

```
(set-face-background-pixmap 'default "/path/to/image.xpm")
(set-face-background-pixmap 'bold    "/path/to/another_image.xpm")
```

and so on. You can also do this interactively via M-x edit-faces .

Q3.2.7: How do I display non-ASCII characters?

If you're using a Mule-enabled XEmacs, then display is automatic. If you're not seeing the characters you expect, either (1) you don't have appropriate fonts available or (2) XEmacs did not correctly detect the coding system (see ⟨undefined⟩ [Recognize Coding], page ⟨undefined⟩). In case (1), install fonts as is customary for your platform. In case (2), you need to tell XEmacs explicitly what coding systems you're using. ⟨undefined⟩ [Specify Coding], page ⟨undefined⟩.

If your XEmacs is not Mule-enabled, and for some reason getting a Mule-enabled XEmacs seems like the wrong thing to do, all is not lost. You can arrange it by brute force. In '`event-Xt.c`' (suppress the urge to look in this file—play Doom instead, because you'll survive longer), it is written:

> In a non-Mule world, a user can still have a multi-lingual editor, by doing `(set-face-font "-*-iso8859-2" (current-buffer))` for all their Latin-2 buffers, etc.

For the related problem of *inputting* non-ASCII characters in a non-Mule XEmacs, See [Q3.5.7], page 38.

3.3: The Modeline

Q3.3.1: How can I make the modeline go away?

```
(set-specifier has-modeline-p nil)
```

Q3.3.2: How do you have XEmacs display the line number in the modeline?

Add the following line to your '`init.el`'/'`.emacs`' file to display the line number:

```
(line-number-mode 1)
```

Use the following to display the column number:

```
(column-number-mode 1)
```

Or select from the `Options` menu
`Advanced (Customize)->Emacs->Editing->Basics->Line Number Mode` and/or
`Advanced (Customize)->Emacs->Editing->Basics->Column Number Mode`

Or type M-x customize ⟨RET⟩ editing-basics ⟨RET⟩.

Q3.3.3: How do I get XEmacs to put the time of day on the modeline?

Add the following line to your '`init.el`'/'`.emacs`' file to display the time:

```
(display-time)
```

See `Customize` from the `Options` menu for customization.

Q3.3.4: How do I turn off current chapter from AUC TeX modeline?

With AUC TeX, fast typing is hard because the current chapter, section etc. are given in the modeline. How can I turn this off?

It's not AUC TeX, it comes from `func-menu` in '`func-menu.el`'.

David Hughes writes:

> Try this; you'll still get the function name displayed in the modeline, but it won't attempt to keep track when you modify the file. To refresh when it gets out of synch, you simply need click on the '`Rescan Buffer`' option in the function-menu.
>
> ```
> (setq-default fume-auto-rescan-buffer-p nil)
> ```

Q3.3.5: How can one change the modeline color based on the mode used?

You can use something like the following:

```
(add-hook 'lisp-mode-hook
          (lambda ()
            (set-face-background 'modeline "red" (current-buffer))))
```

Then, when editing a Lisp file (i.e. when in Lisp mode), the modeline colors change from the default set in your '`init.el`'/'`.emacs`'. The change will only be made in the buffer you just entered (which contains the Lisp file you are editing) and will not affect the modeline colors anywhere else.

Notes:

- The hook is the mode name plus `-hook`. eg. c-mode-hook, c++-mode-hook, emacs-lisp-mode-hook (used for your '`init.el`'/'`.emacs`' or a '`xx.el`' file), lisp-interaction-mode-hook (the '`*scratch*`' buffer), text-mode-hook, etc.
- Be sure to use `add-hook`, not `(setq c-mode-hook xxxx)`, otherwise you will erase anything that anybody has already put on the hook.
- You can also do `(set-face-font 'modeline` font`)`, eg. `(set-face-font 'modeline "*bold-r-normal*140-100-100*" (current-buffer))` if you wish the modeline font to vary based on the current mode.

There are additional modeline faces, `modeline-buffer-id`, `modeline-mousable`, and `modeline-mousable-minor-mode`, which you may want to customize.

3.4: Multiple Device Support

Q3.4.1: How do I open a frame on another screen of my multi-headed display?

Use the command M-x make-frame-on-display. This command is also on the File menu in the menubar.

The command `make-frame-on-tty` also exists, which will establish a connection to any tty-like device. Opening the TTY devices should be left to `gnuclient`, though.

Q3.4.2: Can I really connect to a running XEmacs after calling up over a modem? How?

Yes. Use `gnuclient -nw`. (Prior to 20.3, use the `gnuattach` program supplied with XEmacs instead.)

Also see [Q5.0.12], page 61.

3.5: The Keyboard

Q3.5.1: How can I bind complex functions (or macros) to keys?

As an example, say you want the paste key on a Sun keyboard to insert the current Primary X selection at point. You can accomplish this with:

```
(define-key global-map [f18] 'x-insert-selection)
```

However, this only works if there is a current X selection (the selection will be highlighted). The functionality I like is for the paste key to insert the current X selection if there is one, otherwise insert the contents of the clipboard. To do this you need to pass arguments to `x-insert-selection`. This is done by wrapping the call in a 'lambda form:

```
(global-set-key [f18]
  (lambda () (interactive) (x-insert-selection t nil)))
```

This binds the f18 key to a generic functional object. The interactive spec is required because only interactive functions can be bound to keys.

For the FAQ example you could use:

```
(global-set-key [(control ?.)]
  (lambda () (interactive) (scroll-up 1)))
(global-set-key [(control ?;)]
  (lambda () (interactive) (scroll-up -1)))
```

This is fine if you only need a few functions within the lambda body. If you're doing more it's cleaner to define a separate function as in question 3.5.3 (see [Q3.5.3], page 36).

Q3.5.2: How can I stop down-arrow from adding empty lines to the bottom of my buffers?

Add the following line to your '`init.el`'/'`.emacs`' file:

```
(setq next-line-add-newlines nil)
```

This has been the default setting in XEmacs for some time.

Q3.5.3: How do I bind C-. and C-; to scroll one line up and down?

Add the following (Thanks to Richard Mlynarik and Wayne Newberry) to '`.emacs`':

```
(defun scroll-up-one-line ()
  (interactive)
  (scroll-up 1))

(defun scroll-down-one-line ()
  (interactive)
  (scroll-down 1))

(global-set-key [(control ?.)] 'scroll-up-one-line) ; C-.
(global-set-key [(control ?;)] 'scroll-down-one-line) ; C-;
```

The key point is that you can only bind simple functions to keys; you can not bind a key to a function that you're also passing arguments to. (see [Q3.5.1], page 36 for a better answer).

Q3.5.4: Globally binding Delete ?

I cannot manage to globally bind my Delete key to something other than the default. How does one do this?

Answer: The problem is that many modes explicitly bind Delete . To get around this, try the following:

```
(defun foo ()
  (interactive)
  (message "You hit DELETE"))

(define-key key-translation-map 'delete 'redirected-delete)
(global-set-key 'redirected-delete 'foo)
```

Also see [Q3.5.10], page 39.

Q3.5.5: Scrolling one line at a time.

Can the cursor keys scroll the screen a line at a time, rather than the default half page jump? I tend it to find it disorienting.

Try this:

```
(defun scroll-one-line-up (&optional arg)
  "Scroll the selected window up (forward in the text) one line (or N lines)."
  (interactive "p")
  (scroll-up (or arg 1)))

(defun scroll-one-line-down (&optional arg)
  "Scroll the selected window down (backward in the text) one line (or N)."
  (interactive "p")
  (scroll-down (or arg 1)))

(global-set-key [up]   'scroll-one-line-up)
(global-set-key [down] 'scroll-one-line-down)
```

The following will also work but will affect more than just the cursor keys (i.e. C-n and C-p):

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

Starting with XEmacs-20.3 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Environment->Windows->Scroll Step...` or type M-x customize ⟨RET⟩ windows ⟨RET⟩.

Q3.5.6: How to map Help key alone on Sun type4 keyboard?

The following works in GNU Emacs 19:

```
(global-set-key [help] ’help-command);; Help
```

The following works in XEmacs with the addition of shift:

```
(global-set-key [(shift help)] ’help-command);; Help
```

But it doesn’t work alone. This is in the file ‘`PROBLEMS`’ which should have come with your XEmacs installation: *Emacs ignores the* help *key when running OLWM.*

OLWM grabs the help key, and retransmits it to the appropriate client using `XSendEvent`. Allowing Emacs to react to synthetic events is a security hole, so this is turned off by default. You can enable it by setting the variable `x-allow-sendevents` to t. You can also cause fix this by telling OLWM to not grab the help key, with the null binding `OpenWindows.KeyboardCommand.Help:`.

Q3.5.7: How can you type in special characters in XEmacs?

One way is to use the package `x-compose`. Then you can use sequences like Compose' a to get , etc.

Another way is to use the `iso-insert` package. Then you can use sequences like C-x 8 " a to get , etc.

Glynn Clements writes:

> It depends upon your X server.
>
> Generally, the simplest way is to define a key as Multi_key with xmodmap, e.g.
>
> ```
> xmodmap -e ’keycode 0xff20 = Multi_key’
> ```
>
> You will need to pick an appropriate keycode. Use xev to find out the keycodes for each key.
>
> [NB: On a ‘Windows’ keyboard, recent versions of XFree86 automatically define the right ‘Windows’ key as Multi_key’.]
>
> Once you have Multi_key defined, you can use e.g.
>
> ```
> Multi a ’        =>
> Multi e "        =>
> Multi c ,        =>
> ```
>
> etc.
>
> Also, recent versions of XFree86 define various AltGr-<key> combinations as dead keys, i.e.
>
> ```
> AltGr [          => dead_diaeresis
> AltGr ]          => dead_tilde
> AltGr ;          => dead_acute
> ```
>
> etc.
>
> Running ‘`xmodmap -pk`’ will list all of the defined keysyms.

For the related problem of *displaying* non-ASCII characters in a non-Mule XEmacs, See [Q3.2.7], page 34.

Q3.5.8: [This question intentionally left blank]

Obsolete question, left blank to avoid renumbering.

Q3.5.9: How do I make the Delete key delete forward?

With XEmacs-20.2 use the `delbs` package:

```
(require 'delbs)
```

This will give you the functions `delbs-enable-delete-forward` to set things up, and `delbs-disable-delete-forward` to revert to “normal” behavior. Note that `delbackspace` package is obsolete.

Starting with XEmacs-20.3 better solution is to set variable `delete-key-deletes-forward` to t. You can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Editing->Basics->Delete Key Deletes Forward` or type M-x customize ⟨RET⟩ editing-basics ⟨RET⟩.

Also see [Q3.5.4], page 37.

Q3.5.10: Can I turn on “sticky” modifier keys?

Yes, with `(setq modifier-keys-are-sticky t)`. This will give the effect of being able to press and release Shift and have the next character typed come out in upper case. This will affect all the other modifier keys like Control and Meta as well.

Ben Wing writes:

> One thing about the sticky modifiers is that if you move the mouse out of the frame and back in, it cancels all currently “stuck” modifiers.

Q3.5.11: How do I map the arrow keys?

Say you want to map C-⟨right⟩ to forward-word:

Sam Steingold writes:

```
; both XEmacs and Emacs
(define-key global-map [(control right)] 'forward-word)
```

or

```
; Emacs only
(define-key global-map [C-right] 'forward-word)
```

or

```
; ver > 20, both
(define-key global-map (kbd "C-<right>") 'forward-word)
```

3.6: The Cursor

Q3.6.1: Is there a way to make the bar cursor thicker?

I'd like to have the bar cursor a little thicker, as I tend to "lose" it often.

For a 1 pixel bar cursor, use:

```
(setq bar-cursor t)
```

For a 2 pixel bar cursor, use:

```
(setq bar-cursor 'anything-else)
```

Starting with XEmacs-20.3 you can also change these with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Environment->Display->Bar Cursor...` or type M-x customize RET display RET.

You can use a color to make it stand out better:

```
Emacs*cursorColor:      Red
```

Q3.6.2: Is there a way to get back the block cursor?

```
(setq bar-cursor nil)
```

Starting with XEmacs 20.3 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Environment->Display->Bar Cursor...` or type M-x customize RET display RET.

Q3.6.3: Can I make the cursor blink?

Yes, like this:

```
(blink-cursor-mode)
```

This function toggles between a steady cursor and a blinking cursor. You may also set this mode from the menu bar by selecting '`Options => Frame Appearance => Blinking Cursor`'. Remember to save options.

3.7: The Mouse and Highlighting

Q3.7.1: How can I turn off Mouse pasting?

I keep hitting the middle mouse button by accident and getting stuff pasted into my buffer so how can I turn this off?

Here is an alternative binding, whereby the middle mouse button selects (but does not cut) the expression under the mouse. Clicking middle on a left or right paren will select to the matching one. Note that you can use `define-key` or `global-set-key`.

```
(defun mouse-set-point-and-select (event)
  "Sets the point at the mouse location, then marks following form"
  (interactive "@e")
  (mouse-set-point event)
  (mark-sexp 1))
(define-key global-map [button2] 'mouse-set-point-and-select)
```

Q3.7.2: How do I set control/meta/etc modifiers on mouse buttons?

Use, for instance, `[(meta button1)]`. For example, here is a common setting for Common Lisp programmers who use the bundled `ilisp` package, whereby meta-button1 on a function name will find the file where the function name was defined, and put you at that location in the source file.

[Inside a function that gets called by the lisp-mode-hook and ilisp-mode-hook]

```
(local-set-key [(meta button1)] 'edit-definitions-lisp)
```

Q3.7.3: Clicking the left button does not do anything in buffer list.

I do C-x C-b to get a list of buffers and the entries get highlighted when I move the mouse over them but clicking the left mouse does not do anything.

Use the middle mouse button.

Q3.7.4: How can I get a list of buffers when I hit mouse button 3?

The following code will replace the default popup on button3:

```
(global-set-key [button3] 'popup-buffer-menu)
```

Q3.7.5: Why does cut-and-paste not work between XEmacs and a cmdtool?

We don't know. It's a bug. There does seem to be a work-around, however. Try running xclipboard first. It appears to fix the problem even if you exit it. (This should be mostly fixed in 19.13, but we haven't yet verified that).

Q3.7.6: How I can set XEmacs up so that it pastes where the text cursor is?

By default XEmacs pastes X selections where the mouse pointer is. How do I disable this?

Examine the function `mouse-yank`, by typing C-h f mouse-yank ⟨RET⟩.

To get XEmacs to paste at the text cursor, add this your '`init.el`'/'`.emacs`':

```
(setq mouse-yank-at-point t)
```

Starting with XEmacs-20.2 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Editing->Mouse->Yank At Point...` or type M-x customize ⟨RET⟩ mouse⟨RET⟩.

Q3.7.7: How do I select a rectangular region?

Just select the region normally, then use the rectangle commands (e.g. `kill-rectangle` on it. The region does not highlight as a rectangle, but the commands work just fine.

To actually sweep out rectangular regions with the mouse you can use `mouse-track-do-rectangle` which is assigned to M-button1. Then use rectangle commands.

You can also do the following to change default behavior to sweep out rectangular regions:

```
(setq mouse-track-rectangle-p t)
```

Starting with XEmacs-20.2 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Editing->Mouse->Track Rectangle...` or type `M-x customize RET mouse RET`.

```
mouse-track-do-rectangle: (event)
  -- an interactive compiled Lisp function.
Like 'mouse-track' but selects rectangles instead of regions.
```

Q3.7.8: Why does `M-w` take so long?

It actually doesn't. It leaves the region visible for a second so that you can see what area is being yanked. If you start working, though, it will immediately complete its operation. In other words, it will only delay for a second if you let it.

3.8: The Menubar and Toolbar

Q3.8.1: How do I get rid of the menu (or menubar)?

```
(set-specifier menubar-visible-p nil)
```

Q3.8.2: Can I customize the basic menubar?

For an extensive menubar, add this line to your '`init.el`'/'`.emacs`':

```
(load "big-menubar")
```

If you'd like to write your own, this file provides as good a set of examples as any to start from. The file is located in edit-utils package.

Q3.8.3: How do I control how many buffers are listed in the menu Buffers List ?

Add the following to your '`init.el`'/'`.emacs`' (suit to fit):

```
(setq buffers-menu-max-size 20)
```

For no limit, use an argument of '`nil`'.

Starting with XEmacs-20.3 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Environment->Menu->Buffers Menu->Max Size...` or type `M-x customize RET buffers-menu RET`.

Q3.8.4: Resources like `Emacs*menubar*font` are not working?

I am trying to use a resource like `Emacs*menubar*font` to set the font of the menubar but it's not working.

In Motif, the use of '`font`' resources is obsoleted in order to support internationalization. If you are using the real Motif menubar, this resource is not recognized at all; you have to say:

```
Emacs*menubar*fontList: FONT
```

If you are using the Lucid menubar, for backward compatibility with existing user configurations, the '`font`' resource is recognized. Since this is not supported by Motif itself, the code is a kludge and the '`font`' resource will be recognized only if the '`fontList`' resource resource is unset. This means that the resource

```
*fontList: FONT
```

will override

```
Emacs*menubar*font: FONT
```

even though the latter is more specific.

In non-Motif configurations using '`--with-mule`' and '`--with-xfs`' it *is* necessary to use the `fontSet` resource *instead of* the `font` resource. The backward compatibility kludge was never implemented for non-Motif builds. Example:

```
*fontSet: FONT
```

Q3.8.5: How can I bind a key to a function to toggle the toolbar?

Try something like:

```
(defun my-toggle-toolbar ()
  (interactive)
  (set-specifier default-toolbar-visible-p
                 (not (specifier-instance default-toolbar-visible-p))))
(global-set-key "\C-xT" 'my-toggle-toolbar)
```

There are redisplay bugs in 19.14 that may make the preceding result in a messed-up display, especially for frames with multiple windows. You may need to resize the frame before XEmacs completely realizes the toolbar is really gone.

Thanks to Martin Buchholz for the correct code.

3.9: Scrollbars

Q3.9.1: How can I disable the scrollbar?

To disable them for all frames, add the following line to your '`.Xresources`':

```
Emacs.scrollBarWidth:  0
```

Or select from the `Options` menu `Frame Appearance->Scrollbars`. Remember to save options.

To turn the scrollbar off on a per-frame basis, use the following function:

```
(set-specifier scrollbar-width 0 (selected-frame))
```

You can actually turn the scrollbars on at any level you want by substituting for (selected-frame) in the above command. For example, to turn the scrollbars off only in a single buffer:

```
(set-specifier scrollbar-width 0 (current-buffer))
```

Q3.9.2: How can one use resources to change scrollbar colors?

Here's a recap of how to use resources to change your scrollbar colors:

```
! Motif scrollbars

Emacs*XmScrollBar.Background: skyblue
Emacs*XmScrollBar.troughColor: lightgray

! Athena scrollbars

Emacs*Scrollbar.Foreground: skyblue
Emacs*Scrollbar.Background: lightgray
```

Note the capitalization of `Scrollbar` for the Athena widget.

Q3.9.3: Moving the scrollbar can move the point; can I disable this?

When I move the scrollbar in an XEmacs window, it moves the point as well, which should not be the default behavior. Is this a bug or a feature? Can I disable it?

The current behavior is a feature, not a bug. Point remains at the same buffer position as long as that position does not scroll off the screen. In that event, point will end up in either the upper-left or lower-left hand corner.

This cannot be changed.

Q3.9.4: How can I turn off automatic horizontal scrolling in specific modes?

Do `(setq truncate-lines t)` in the mode-hooks for any modes in which you want lines truncated.

More precisely: If `truncate-lines` is nil, horizontal scrollbars will never appear. Otherwise, they will appear only if the value of `scrollbar-height` for that buffer/window/etc. is non-zero. If you do

```
(set-specifier scrollbar-height 0)
```

then horizontal scrollbars will not appear in truncated buffers unless the package specifically asked for them.

3.10: Text Selections

Q3.10.1: How can I turn off or change highlighted selections?

The `zmacs` mode allows for what some might call gratuitous highlighting for selected regions (either by setting mark or by using the mouse). This is the default behavior. To turn off, add the following line to your '`init.el`'/'`.emacs`' file:

```
(setq zmacs-regions nil)
```

Starting with XEmacs-20.2 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Editing->Basics->Zmacs Regions` or type M-x customize ⟨RET⟩ editing-basics ⟨RET⟩.

To change the face for selection, look at `Options->Customize` on the menubar.

Q3.10.2: How do I get that typing on an active region removes it?

I want to change things so that if I select some text and start typing, the typed text replaces the selected text, similar to Motif.

You want to use something called pending delete Pending delete is what happens when you select a region (with the mouse or keyboard) and you press a key to replace the selected region by the key you typed. Usually backspace kills the selected region.

To get this behavior, add the following lines to your '`init.el`'/'`.emacs`':

```
(cond
 ((fboundp 'turn-on-pending-delete)
  (turn-on-pending-delete))
 ((fboundp 'pending-delete-on)
  (pending-delete-on t)))
```

Note that this will work with both Backspace and Delete. This code is a tad more complicated than it has to be for XEmacs in order to make it more portable.

Q3.10.3: Can I turn off the highlight during isearch?

I do not like my text highlighted while I am doing isearch as I am not able to see what's underneath. How do I turn it off?

Put the following in your '`init.el`'/'`.emacs`':

```
(setq isearch-highlight nil)
```

Starting with XEmacs-20.2 you can also change this with Customize. Type M-x customize-variable ⟨RET⟩ isearch-highlight ⟨RET⟩.

Note also that isearch-highlight affects query-replace and ispell. Instead of disabling isearch-highlight you may find that a better solution consists of customizing the `isearch` face.

Q3.10.4: How do I turn off highlighting after C-x C-p (mark-page)?

Put this in your `.emacs`:

```
(setq zmacs-regions nil)
```

Warning: This command turns off all region highlighting.

Also see [Q3.10.1], page 44.

Q3.10.5: The region disappears when I hit the end of buffer while scrolling.

This has been fixed by default starting with XEmacs-20.3.

With older versions you can turn this feature (if it indeed is a feature) off like this:

```
(defadvice scroll-up (around scroll-up freeze)
  (interactive "_P")
  (let ((zmacs-region-stays t))
    (if (interactive-p)
        (condition-case nil
            ad-do-it
          (end-of-buffer (goto-char (point-max))))
      ad-do-it)))

(defadvice scroll-down (around scroll-down freeze)
  (interactive "_P")
  (let ((zmacs-region-stays t))
    (if (interactive-p)
        (condition-case nil
            ad-do-it
          (beginning-of-buffer (goto-char (point-min))))
      ad-do-it)))
```

Thanks to T. V. Raman for assistance in deriving this answer.

Q3.10.6: Why is killing so slow?

This actually is an X Windows question, although you'll notice it with keyboard operations as well as while using the GUI. Basically, there are four ways to communicate interprogram via the X server:

Primary selection	a transient selection that gets replaced every time a new selection is made
Secondary selection	for "exchanging" with the primary selection
Cut buffers	a clipboard internal to the X server (deprecated)
Clipboard selection	a selection with a notification protocol that allows a separate app to manage the clipboard

The cut buffers are deprecated because managing them is even more inefficient than the clipboard notification protocol. The primary selection works fine for many users and applications, but is not very robust under intensive or sophisticated use.

In Motif and MS Windows, a clipboard has become the primary means for managing cut and paste. These means that "modern" applications tend to be oriented toward a true clipboard, rather than the primary selection. (On Windows, there is nothing equivalent to the primary selection.) It's not that XEmacs doesn't support the simple primary selection method, it's that more and more other applications don't.

So the slowdown occurs because XEmacs now engages in the clipboard notification protocol on *every* kill. This is especially slow on Motif.

With most people running most clients and server on the same host, and many of the rest working over very fast communication, you may expect that the situation is not going to improve.

There are a number of workarounds. The most effective is to use a special command to do selection ownership only when you intend to paste to another application. Useful commands are `kill-primary-selection` and `copy-primary-selection`. These work only on text selected with the mouse (probably; experiment), and are bound by default to the Cut and Copy, respectively, buttons on the toolbar. `copy-primary-selection` is also bound to C-Insert . You can yank the clipboard contents with `yank-primary-selection`, bound to the Paste toolbar button and Sh-Insert .

If you are communicating by cut and paste with applications that use the primary selection, then you can customize `interprogram-cut-function` to `nil`, restoring the XEmacs version 20 behavior. How can you tell if a program will support this? Motifly-correct programs require the clipboard; you lose. For others, only by trying it. You also need to customize the complementary `interprogram-paste-function` to `nil`. (Otherwise XEmacs-to-XEmacs pastes will not work correctly.)

You may get some relief on Motif by setting `x-selection-strict-motif-ownership` to nil, but this means you will only intermittently be able to paste XEmacs kills to Motif applications.

Thanks to Jeff Mincy and Glynn Clements for corrections.

4 Major Subsystems

This is part 4 of the XEmacs Frequently Asked Questions list. This section is devoted to major XEmacs subsystems.

4.0: Reading Mail with VM

Q4.0.1: How do I set up VM to retrieve mail from a remote site using POP?

Use `vm-spool-files`, like this for example:

```
(setq vm-spool-files '("/var/spool/mail/wing"
                       "netcom23.netcom.com:110:pass:wing:MYPASS"))
```

Of course substitute your actual password for MYPASS.

Q4.0.2: How do I get VM to filter mail for me?

One possibility is to use procmail to split your mail before it gets to VM. I prefer this personally, since there are many strange and wonderful things one can do with procmail. Procmail may be found at `ftp://ftp.informatik.rwth-aachen.de/pub/packages/procmail/`.

Also see the Mail Filtering FAQ at:
`ftp://rtfm.mit.edu/pub/usenet/news.answers/mail/filtering-faq`.

Q4.0.3: How can I get VM to automatically check for new mail?

John Turner writes:

> Use the following:
>
> ```
> (setq vm-auto-get-new-mail 60)
> ```

Q4.0.4: [This question intentionally left blank]

Obsolete question, left blank to avoid renumbering.

Q4.0.5: How do I get my outgoing mail archived?

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

Q4.0.6: I have various addresses at which I receive mail. How can I tell VM to ignore them when doing a "reply-all"?

Set `vm-reply-ignored-addresses` to a list, like

```
(setq vm-reply-ignored-addresses
      '("wing@nuspl@nvwls.cc.purdue.edu,netcom[0-9]*.netcom.com"
        "wing@netcom.com" "wing@xemacs.org"))
```

Note that each string is a regular expression.

Q4.0.7: Is there a mailing list or FAQ for VM?

A FAQ for VM exists at `http://www.wonderworks.com/vm/FAQ.html`.

VM has its own newsgroups gnu.emacs.vm.info and gnu.emacs.vm.bug.

Q4.0.8: Remote mail reading with VM.

My mailbox lives at the office on a big honkin server. My regular INBOX lives on my honkin desktop machine. I now can PPP to the office from home which is far from honking... I'd like to be able to read mail at home without storing it here and I'd like to use xemacs and VM at home... Is there a recommended setup?

Joseph J. Nuspl Jr. writes:

> There are several ways to do this.
>
> 1. Set your display to your home machine and run dxpc or one of the other X compressors.
> 2. NFS mount your desktop machine on your home machine and modify your pop command on your home machine to rsh to your desktop machine and actually do the pop get's.
> 3. Run a POP server on your desktop machine as well and do a sort of two tiered POP get.

William Perry adds:

> Or you could run a pop script periodically on your desktop machine, and just use ange-ftp or NFS to get to your mailbox. I used to do this all the time back at IU.

Q4.0.9: rmail or VM gets an error incorporating new mail.

Quoting the XEmacs PROBLEMS file:

> rmail and VM get new mail from '`/usr/spool/mail/$USER`' using a program called `movemail`. This program interlocks with `/bin/mail` using the protocol defined by `/bin/mail`.
>
> There are two different protocols in general use. One of them uses the `flock` system call. The other involves creating a lock file; `movemail` must be able to write in '`/usr/spool/mail`' in order to do this. You control which one is used by defining, or not defining, the macro `MAIL_USE_FLOCK` in '`config.h`' or the m- or s- file it includes.
>
> IF YOU DON'T USE THE FORM OF INTERLOCKING THAT IS NORMAL ON YOUR SYSTEM, YOU CAN LOSE MAIL!
>
> If your system uses the lock file protocol, and fascist restrictions prevent ordinary users from writing the lock files in '`/usr/spool/mail`', you may need to make `movemail` setgid to a suitable group such as '`mail`'. You can use these commands (as root):
>
> ```
> chgrp mail movemail
> chmod 2755 movemail
> ```
>
> If your system uses the lock file protocol, and fascist restrictions prevent ordinary users from writing the lock files in '`/usr/spool/mail`', you may need to make `movemail` setgid to a suitable group such as `mail`. To do this, use the following commands (as root) after doing the make install.
>
> ```
> chgrp mail movemail
> chmod 2755 movemail
> ```
>
> Installation normally copies movemail from the build directory to an installation directory which is usually under '`/usr/local/lib`'. The installed copy of `movemail` is usually in the directory '`/usr/local/lib/emacs/VERSION/TARGET`'. You must change the group and mode of the installed copy; changing the group and mode of the build directory copy is ineffective.

Q4.0.10: How do I make VM stay in a single frame?

John.John S Cooper writes:

```
                                   ; Don't use multiple frames
(setq vm-frame-per-composition nil)
(setq vm-frame-per-folder nil)
(setq vm-frame-per-edit nil)
(setq vm-frame-per-summary nil)
```

Q4.0.11: How do I make VM or mh-e display graphical smilies?

For mh-e use the following:

```
(add-hook 'mh-show-mode-hook '(lambda ()
                                (smiley-region (point-min)
                                               (point-max))))
```

WJCarpenter writes: For VM use the following:

```
              (autoload 'smiley-region "smiley" nil t)
              (add-hook 'vm-select-message-hook
                        '(lambda ()
                           (smiley-region (point-min)
                                          (point-max))))
```

For tm use the following:

```
(autoload 'smiley-buffer "smiley" nil t)
(add-hook 'mime-viewer/plain-text-preview-hook 'smiley-buffer)
```

Q4.0.12: Customization of VM not covered in the manual, or here.

giacomo boffi writes:

> The meta-answer is to look into the file ‘`vm-vars.el`’, in the vm directory of the lisp library.
>
> ‘`vm-vars.el`’ contains, initializes and carefully describes, with examples of usage, the plethora of user options that *fully* control VM’s behavior.
>
> Enter vm-vars, `forward-search` for toolbar, find the variables that control the toolbar placement, appearance, existence, copy to your ‘`init.el`’/‘`.emacs`’ or ‘`.vm`’ and modify according to the detailed instructions.
>
> The above also applies to all the various features of VM: search for some keywords, maybe the first you conjure isn’t appropriate, find the appropriate variables, copy and experiment.

4.1: Web browsing with W3

Q4.1.1: What is W3?

W3 is an advanced graphical browser written in Emacs lisp that runs on XEmacs. It has full support for cascaded style sheets, and more...

It has a home web page at `http://www.cs.indiana.edu/elisp/w3/docs.html`.

Q4.1.2: How do I run W3 from behind a firewall?

There is a long, well-written, detailed section in the W3 manual that describes how to do this. Look in the section entitled "Firewalls".

Q4.1.3: Is it true that W3 supports style sheets and tables?

Yes, and much more. W3, as distributed with the latest XEmacs is a full-featured web browser.

4.2: Reading Netnews and Mail with Gnus

Q4.2.1: GNUS, (ding) Gnus, Gnus 5, September Gnus, Red Gnus, Quassia Gnus, argh!

The Gnus numbering issues are not meant for mere mortals to know them. If you feel you *must* enter the muddy waters of Gnus, visit the excellent FAQ, maintained by Justin Sheehy, at:

`http://www.ccs.neu.edu/software/contrib/gnus/`

See also Gnus home page

`http://www.gnus.org/`

Q4.2.2: This question intentionally left blank.

Obsolete question, left blank to avoid renumbering.

Q4.2.3: How do I make Gnus stay within a single frame?

The toolbar code to start Gnus opens the new frame—and it's a feature rather than a bug. If you don't like it, but would still like to click on the seemly icon, use the following code:

```
(defun toolbar-news ()
  (gnus))
```

It will redefine the callback function of the icon to just call `gnus`, without all the fancy frame stuff.

Q4.2.4: How do I customize the From: line?

How do I change the `From:` line? I have set gnus-user-from-line to

```
Gail Gurman <gail.gurman@sybase.com>
```

, but XEmacs Gnus doesn't use it. Instead it uses

`Gail Mara Gurman gailg@deall`

and then complains that it's incorrect. Also, as you perhaps can see, my Message-ID is screwy. How can I change that?

Lars Magne Ingebrigtsen writes:

> Set `user-mail-address` to '`gail.gurman@sybase.com`' or `mail-host-address` to '`sybase.com`'.

4.3: Other Mail & News

Q4.3.1: How can I read and/or compose MIME messages?

VM supports MIME natively.

You probably want to use the Tools for MIME (tm). See [Q4.3.2], page 52, for details.

Trey Jackson has an Emacs & MIME web page at `http://bmrc.berkeley.edu/~trey/emacs/mime.html`.

Another possibility is RMIME. You may find RMIME at `http://www.cinti.net/~rmoody/rmime/index.html`.

Q4.3.2: What is TM and where do I get it?

TM stands for Tools for MIME and not Tiny MIME. TM integrates with all major XEmacs packages like Gnus (all flavors), VM, MH-E, and mailcrypt. It provides totally transparent and trouble-free MIME support. When appropriate a message will be decoded in place in an XEmacs buffer.

TM now comes as a package with XEmacs 19.16 and XEmacs 20.2.

TM was written by MORIOKA Tomohiko and KOBAYASHI Shuhei.

It is based on the work of UMEDA Masanobu, the original writer of GNUS.

The following information is from the ‘`README`’:

tm is a MIME package for GNU Emacs. tm has following functions:

- MIME style multilingual header.
- MIME message viewer (mime/viewer-mode).
- MIME message composer (mime/editor-mode).
- MIME extenders for mh-e, GNUS, RMAIL and VM.

tm is available from following anonymous ftp sites:

- `ftp://ftp.unicamp.br/pub/mail/mime/tm/` (Brasil).
- `ftp://ftp.th-darmstadt.de/pub/editors/GNU-Emacs/lisp/mime/` (Germany).
- `ftp://ftp.tnt.uni-hannover.de/pub/editors/xemacs/contrib/` (Germany).

Don’t let the installation procedure & instructions stop you from trying this package out—it’s much simpler than it looks, and once installed, trivial to use.

Steve Youngs writes:

> All the major Emacs Lisp based MUAs (Gnus, MH-E, and VM) all do their own thing when it comes to MIME so you won’t need TM to get MIME support in these packages.

Q4.3.3: Why isn't this movemail program working?

Ben Wing `ben@xemacs.org` writes:

> It wasn’t chown’ed/chmod’d correctly.

Q4.3.4: Movemail is also distributed by Netscape? Can that cause problems?

Steve Baur writes:

> Yes. Always use the movemail installed with your XEmacs. Failure to do so can result in lost mail.

Please refer to Jamie Zawinski's notes at `http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/movemail.html`. In particular, this document will show you how to make Netscape use the version of movemail configured for your system by the person who built XEmacs.

Q4.3.5: Where do I find pstogif (required by tm)?

pstogif is part of the latex2html package.

Jan Vroonhof writes:

latex2html is best found at the CTAN hosts and their mirrors in ‘`tex-archive/support/latex2html`’.

CTAN hosts are:

- `ftp://ftp.tex.ac.uk/tex-archive/support/latex2html/`.
- `ftp://ftp.dante.de/tex-archive/support/latex2html/`.

There is a good mirror at ftp.cdrom.com; `ftp://ftp.cdrom.com/pub/tex/ctan/support/latex2html/`.

4.4: Sparcworks, EOS, and WorkShop

Q4.4.1: What is SPARCworks, EOS, and WorkShop?

John Turner writes:

> SPARCworks is SunSoft's development environment, comprising compilers (C, C++, FORTRAN 77, Fortran 90, Ada, and Pascal), a debugger, and other tools such as TeamWare (for configuration management), MakeTool, etc.

See `http://www.sun.com/software/Developer-products/` for more info.

EOS stands for "Era on SPARCworks", but I don't know what Era stands for.

EOS is the integration of XEmacs with the SPARCworks debugger. It allows one to use an XEmacs frame to view code (complete with fontification, etc.), set breakpoints, print variables, etc., while using the SPARCworks debugger. It works very well and I use it all the time.

Chuck Thompson writes:

> Era stood for "Emacs Rewritten Again". It was what we were calling the modified version of Lucid Emacs for Sun when I was dragged, er, allowed to work on this wonderful editor.

Martin Buchholz writes:

> EOS is being replaced with a new graphical development environment called Sun WorkShop, which is currently (07/96) in Alpha Test. For more details, check out `http://www.sun.com/software/Products/Developer-products`.

Q4.4.2: How do I start the Sun Workshop support in XEmacs 21?

Add the switch —with-workshop to the configure command when building XEmacs and put the following in one of your startup files (e.g. site-start.el or .emacs):

```
(when (featurep 'tooltalk)
  (load "tooltalk-macros")
  (load "tooltalk-util")
  (load "tooltalk-init"))
(when (featurep 'sparcworks)
  (load "sunpro-init")
  (load "ring")
  (load "comint")
  (load "annotations")
  (sunpro-startup))
```

If you are not using the latest Workshop (5.0) you have to apply the following patch:

```
— /opt/SUNWspro/lib/eserve.el.ORIG    Fri May 14 15:23:26 1999
+++ /opt/SUNWspro/lib/eserve.el Fri May 14 15:24:54 1999
@@ -42,7 +42,7 @@
 (defvar running-xemacs nil "t if we’re running XEmacs")
 (defvar running-emacs  nil "t if we’re running GNU Emacs 19")

-(if (string-match "^\\(19\\|20\\)\..*\\(XEmacs\\|Lucid\\)" emacs-version)
+(if (string-match "\\(XEmacs\\|Lucid\\)" emacs-version)
     (setq running-xemacs t)
     (setq running-emacs  t))
```

4.5: Energize

Q4.5.1: What is/was Energize?

David N Gray writes:

> The files in ‘`lisp/energize`’ are to enable Emacs to interface with the "Energize Programming System", a C and C++ development environment, which was a product of Lucid, Inc. Tragically, Lucid went out of business in 1994, so although Energize is still a great system, if you don’t already have it, there isn’t any way to get it now. (Unless you happen to be in Japan; INS Engineering may still be selling it there. Tartan bought the rights to sell it in the rest of the world, but never did so.)

4.6: Infodock

Q4.6.1: What is Infodock?

InfoDock is an integrated productivity toolset, mainly aimed at technical people, hosted at SourceForge.

InfoDock is built atop the XEmacs variant of GNU Emacs and so has all of the power of Emacs, but with an easier to use and more comprehensive menu-based user interface. The bottom portion of this text describes how it differs from XEmacs and GNU Emacs from the Free Software Foundation.

InfoDock is aimed at people who want a free, turn-key productivity environment. Although InfoDock is customizable, it is not intended for people who like basic versions of Emacs which need to be customized extensively for local use; standard Emacs distributions are better for such uses. InfoDock is for those people who want a complete, pre-customized environment in one package, which they need not touch more than once or twice a year to update to new revisions.

InfoDock is pre-built for SPARC SunOS/Solaris systems, PA-RISC HP-UX, and Intel Linux systems. It is intended for use on a color display, although most features will work on monochrome monitors. Simply unpack InfoDock according to the instructions in the ID-INSTALL file and you are ready to run.

The InfoDock Manual is concise, yet sufficient as a user guide for users who have never used an Emacs-type editor before. For users who are already familiar with Emacs, it supplements the information in the GNU Emacs Manual.

InfoDock menus are much more extensive and more mature than standard Emacs menus. Each menu offers a '`Manual`' item which displays documentation associated with the menu's functions.

Four types of menubars are provided:

1. An extensive menubar providing access to global InfoDock commands.
2. Mode-specific menubars tailored to the current major mode.
3. A simple menubar for basic editing to help novices get started with InfoDock.
4. The standard XEmacs menubar.

Most modes also include mode-specific popup menus. Additionally, region and rectangle popup menus are included.

'`Hyperbole`', the everyday information manager, is a core part of InfoDock. This provides context-sensitive mouse keys, a rolodex-type contact manager, programmable hypertext buttons, and an autonumbered outliner with embedded hyperlink anchors.

The '`OO-Browser`', a multi-language object-oriented code browser, is a standard part of InfoDock.

InfoDock saves a more extensive set of user options than other Emacs versions.

InfoDock inserts a useful file header in many file types, showing the author, summary, and last modification time of each file. A summary program can then be used to summarize all of the files in a directory, for easy MANIFEST file creation.

Your working set of buffers is automatically saved and restored (if you answer yes to a prompt) between InfoDock sessions.

Refined color choices for code highlighting are provided for both dark and light background display frames.

The `C-z` key prefix performs frame-based commands which parallel the `C-x` key prefix for window-based commands.

The Smart Menu system is included for producing command menus on dumb terminals.

Lisp libraries are better categorized according to function.

Extensions and improvements to many areas of Emacs are included, such as: paragraph filling, mail reading with Rmail, shell handling, outlining, code highlighting and browsing, and man page browsing.

InfoDock questions, answers and discussion should go to the mail list `infodock@infodock.com`. Use `infodock-request@infodock.com` to be added or removed from the list. Always include your InfoDock version number when sending help requests.

InfoDock is available across the Internet via anonymous FTP. To get it, first move to a directory into which you want the InfoDock archive files placed. We will call this <DIST-DIR>.

```
cd <DIST-DIR>
```

Ftp to ftp.xemacs.org (Internet Host ID = 128.174.252.16):

```
prompt> ftp ftp.xemacs.org
```

Login as '`anonymous`' with your own <user-id>@<site-name> as a password.

```
Name (ftp.xemacs.org): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: -<your-user-id>@<your-domain>
230 Guest login ok, access restrictions apply.
```

Move to the location of the InfoDock archives:

```
ftp> cd pub/infodock
```

Set your transfer mode to binary:

```
ftp> bin
200 Type set to I.
```

Turn off prompting:

```
ftp> prompt
Interactive mode off.
```

Retrieve the InfoDock archives that you want, either by using a '`get <file>`' for each file you want or by using the following to get a complete distribution, including all binaries:

```
ftp> mget ID-INSTALL
ftp> mget id-*
```

Close the FTP connection:

```
ftp> quit
221 Goodbye.
```

Read the '`ID-INSTALL`' file which you just retrieved for step-by-step installation instructions.

4.7: Other Unbundled Packages

Q4.7.1: What is AUC TeX? Where do you get it?

AUC TeX is a package written by Per Abrahamsen. Starting with XEmacs 19.16, AUC TeX is bundled with XEmacs. The following information is from the '`README`' and website.

AUC TeX is an extensible package that supports writing and formatting TeX files for most variants of GNU Emacs. Many different macro packages are supported, including AMS TeX, LaTeX, and TeXinfo.

The most recent version is always available by ftp at `ftp://sunsite.dk/packages/auctex/auctex.tar.gz`.

In case you don't have access to anonymous ftp, you can get it by an email request to `ftpmail@decwrl.dec.com`.

WWW users may want to check out the AUC TeX page at `http://sunsite.dk/auctex/`.

Q4.7.2: Are there any Emacs Lisp Spreadsheets?

Yes. Check out dismal (which stands for Dis' Mode Ain't Lotus) at `ftp://cs.nyu.edu/pub/local/fox/dismal/`.

Q4.7.3: [This question intentionally left blank]

Q4.7.4: Problems installing AUC TeX.

Jan Vroonhof writes:

> AUC TeX works fine on both stock Emacs and XEmacs has been doing so for a very very long time. This is mostly due to the work of Per Abrahamsen (clap clap) in particular his '`easymenu`' package. Which leads to what is probably the problem...

Most problems with AUC TeX are one of two things:

- The TeX-lisp-directory in '`tex-site.el`' and the makefile don't match.

  Fix: make sure you configure AUC TeX properly **before** installing.
- You have an old version of easymenu.el in your path.

  Fix: use `locate-library` and remove old versions to make sure it **only** finds the one that came with XEmacs.

Q4.7.5: Is there a reason for an Emacs package not to be included in XEmacs?

The reason for an Emacs package not to be included in XEmacs is usually one or more of the following:

1. The package has not been ported to XEmacs. This will typically happen when it uses GNU-Emacs-specific features, which make it fail under XEmacs.

   Porting a package to XEmacs can range from a trivial amount of change to a partial or full rewrite. Fortunately, the authors of modern packages usually choose to support both Emacsen themselves.
2. The package has been decided not to be appropriate for XEmacs. It may have an equivalent or better replacement within XEmacs, in which case the developers may choose not to burden themselves with supporting an additional package.

Each package bundled with XEmacs means more work for the maintainers, whether they want it or not. If you are ready to take over the maintenance responsibilities for the package you port, be sure to say so—we will more likely include it.

3. The package simply hasn't been noted by the XEmacs development. If that's the case, the messages like yours are very useful for attracting our attention.
4. The package was noted by the developers, but they simply haven't yet gotten around to including/porting it. Wait for the next release or, even better, offer your help. It will be gladly accepted and appreciated.

Q4.7.5: Is there a MatLab mode?

Yes, a matlab mode and other items are available at the MathWorks' emacs_add_ons ftp directory.

Q4.7.7: Can I edit files on other hosts?

Yes. Of course XEmacs can use any network file system (such as NFS or Windows file sharing) you have available, and includes some optimizations and safety features appropriate to those environments.

It is also possible to transparently edit files via FTP, ssh, or rsh. That is, XEmacs makes a local copy using the transport in the background, and automatically refreshes the remote original from that copy when you save it. XEmacs also is capable of doing file system manipulations like creating and removing directories and files. The FTP interface is provided by the standard '`efs`' package [Top], page 1. The ssh/rsh interface is provided by the optional '`tramp`' package [Top], page 1.

5 The Miscellaneous Stuff

This is part 5 of the XEmacs Frequently Asked Questions list. This section is devoted to anything that doesn't fit neatly into the other sections.

5.0: Major & Minor Modes

Q5.0.1: How can I do source code highlighting using font-lock?

For most modes, font-lock is already set up and just needs to be turned on. This can be done by adding the line:

```
(require 'font-lock)
```

to your '`init.el`'/'`.emacs`'. (You can turn it on for the current buffer and session only by M-x font-lock-mode .) See the file '`etc/sample.init.el`' ('`etc/sample.emacs`' in XEmacs versions prior to 21.4) for more information.

See also `Syntax Highlighting` from the `Options` menu. Remember to save options.

Q5.0.2: I do not like cc-mode. How do I use the old c-mode?

Well, first off, consider if you really want to do this. cc-mode is much more powerful than the old c-mode. If you're having trouble getting your old offsets to work, try using `c-set-offset` instead. You might also consider using the package `cc-compat`.

But, if you still insist, add the following lines to your ‘`init.el`’/‘`.emacs`’:

```
(fmakunbound 'c-mode)
(makunbound 'c-mode-map)
(fmakunbound 'c++-mode)
(makunbound 'c++-mode-map)
(makunbound 'c-style-alist)
(load-library "old-c-mode")
(load-library "old-c++-mode")
```

This must be done before any other reference is made to either c-mode or c++-mode.

Q5.0.3: How do I get ‘More’ Syntax Highlighting on by default?

Use the following code in your ‘`init.el`’/‘`.emacs`’:

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

See also `Syntax Highlighting` from the `Options` menu. Remember to save options.

Q5.0.4: How can I enable auto-indent and/or Filladapt?

Put the following line in your ‘`init.el`’/‘`.emacs`’:

```
(setq indent-line-function 'indent-relative-maybe)
```

If you want to get fancy, try the `filladapt` package available standard with XEmacs. Put this into your ‘`init.el`’/‘`.emacs`’:

```
(require 'filladapt)
(setq-default filladapt-mode t)
(add-hook 'c-mode-hook 'turn-off-filladapt-mode)
```

This will enable Filladapt for all modes except C mode, where it doesn't work well. To turn Filladapt on only in particular major modes, remove the `(setq-default ...)` line and use `turn-on-filladapt-mode`, like this:

```
(add-hook 'text-mode-hook 'turn-on-filladapt-mode)
```

You can customize filling and adaptive filling with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Editing->Fill->Fill...` or type M-x customize ⟨RET⟩ fill ⟨RET⟩.

Note that well-behaving text-lookalike modes will run `text-mode-hook` by default (e.g. that's what Message does). For the nasty ones, you'll have to provide the `add-hooks` yourself.

Please note that the `fa-extras` package is no longer useful.

Q5.0.5: How can I get XEmacs to come up in text/auto-fill mode by default?

Try the following lisp in your ‘`init.el`’/‘`.emacs`’:

```
(setq default-major-mode 'text-mode)
(setq text-mode-hook 'turn-on-auto-fill)
```

WARNING : note that changing the value of `default-major-mode` from `fundamental-mode` can break a large amount of built-in code that expects newly created buffers to be in `fundamental-mode`. (Changing from `fundamental-mode` to `text-mode` might not wreak too much havoc, but changing to something more exotic like a lisp-mode would break many Emacs packages).

Note that Emacs by default starts up in buffer `*scratch*` in `initial-major-mode`, which defaults to `lisp-interaction-mode`. Thus adding the following form to your Emacs init file will cause the initial `*scratch*` buffer to be put into auto-fill'ed `text-mode`:

```
(setq initial-major-mode
      (lambda ()
        (text-mode)
        (turn-on-auto-fill)))
```

Note that after your init file is loaded, if `inhibit-startup-message` is `nil` (the default) and the startup buffer is `*scratch*` then the startup message will be inserted into `*scratch*`; it will be removed after a timeout by erasing the entire `*scratch*` buffer. Keep in mind this default usage of `*scratch*` if you desire any prior manipulation of `*scratch*` from within your Emacs init file. In particular, anything you insert into `*scratch*` from your init file will be later erased. Also, if you change the mode of the `*scratch*` buffer, be sure that this will not interfere with possible later insertion of the startup message (e.g. if you put `*scratch*` into a nonstandard mode that has automatic font lock rules, then the startup message might get fontified in a strange foreign manner, e.g. as code in some programming language).

Q5.0.6: How do I start up a second shell buffer?

In the `*shell*` buffer:

```
M-x rename-buffer RET *shell-1* RET
M-x shell RET
```

This will then start a second shell. The key is that no buffer named ‘`*shell*`’ can exist. It might be preferable to use M-x rename-uniquely to rename the `*shell*` buffer instead of M-x rename-buffer .

Alternately, you can set the variable `shell-multiple-shells`. If the value of this variable is non-nil, each time shell mode is invoked, a new shell is made

Q5.0.7: Telnet from shell filters too much

I'm using the Emacs M-x shell function, and I would like to invoke and use a telnet session within it. Everything works fine except that now all ‘`^M`’'s are filtered out by Emacs. Fixes?

Use M-x rsh or M-x telnet to open remote sessions rather than doing rsh or telnet within the local shell buffer. Starting with XEmacs-20.3 you can also use M-x ssh to open secure remote session if you have `ssh` installed.

Q5.0.8: Why does edt emulation not work?

We don't know, but you can use tpu-edt emulation instead, which works fine and is a little fancier than the standard edt emulation. To do this, add the following line to your '`init.el`'/'`.emacs`':

```
(tpu-edt)
```

If you don't want it to replace C-h with an edt-style help menu add this as well:

```
(global-set-key [(control h)] 'help-for-help)
```

Q5.0.9: How can I emulate VI and use it as my default mode?

Our recommended VI emulator is viper. To make viper-mode the default, add this to your '`init.el`'/'`.emacs`':

```
(viper-mode)
```

Michael Kifer writes:

> This should be added as close to the top of '`init.el`'/'`.emacs`' as you can get it, otherwise some minor modes may not get viper-ized.

Q5.0.10: [This question intentionally left blank]

Obsolete question, left blank to avoid renumbering

Q5.0.11: [This question intentionally left blank]

Obsolete question, left blank to avoid renumbering

Q5.0.12: How do I disable gnuserv from opening a new frame?

If you set the `gnuserv-frame` variable to the frame that should be used to display buffers that are pulled up, a new frame will not be created. For example, you could put

```
(setq gnuserv-frame (selected-frame))
```

early on in your '`init.el`'/'`.emacs`', to ensure that the first frame created is the one used for your gnuserv buffers.

There is an option to set the gnuserv target to the current frame. See `Options->Display->"Other Window" Location->Make Current Frame Gnuserv Target`

Starting with XEmacs-20.3 you can also change this with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Environment->Gnuserv->Gnuserv Frame...` or type M-x customize ⟨RET⟩ gnuserv ⟨RET⟩.

Q5.0.13: How do I start gnuserv so that each subsequent XEmacs is a client?

Put the following in your '`init.el`'/'`.emacs`' file to start the server:

```
(gnuserv-start)
```

Start your first XEmacs as usual. After that, you can do:

```
gnuclient randomfilename
```

from the command line to get your existing XEmacs process to open a new frame and visit randomfilename in that window. When you're done editing randomfilename, hit C-x # to kill the buffer and get rid of the frame.

See also man page of gnuclient.

Q5.0.14: Strange things are happening in Shell Mode.

Sometimes (i.e. it's not repeatable, and I can't work out why it happens) when I'm typing into shell mode, I hit return and only a portion of the command is given to the shell, and a blank prompt is returned. If I hit return again, the rest of the previous command is given to the shell.

Martin Buchholz writes:

> There is a known problem with interaction between `csh` and the `filec` option and XEmacs. You should add the following to your '`.cshrc`':
>
> ```
> if ( "$TERM" == emacs || "$TERM" == unknown ) unset filec
> ```

Q5.0.15: Where do I get the latest CC Mode?

Barry A. Warsaw writes:

> This can be had from `http://www.python.org/emacs/`.

Q5.0.16: I find auto-show-mode disconcerting. How do I turn it off?

`auto-show-mode` controls whether or not a horizontal scrollbar magically appears when a line is too long to be displayed. This is enabled by default. To turn it off, put the following in your '`init.el`'/'`.emacs`':

```
(setq auto-show-mode nil)
(setq-default auto-show-mode nil)
```

Q5.0.17: How can I get two instances of info?

Before 21.4, you can't. The `info` package does not provide for multiple info buffers. In 21.4, this should be fixed. #### how?

Q5.0.18: [This question intentionally left blank]

Q5.0.19: Is there something better than LaTeX mode?

David Kastrup writes:

> The standard TeX modes leave much to be desired, and are somewhat leniently maintained. Serious TeX users use AUC TeX (see [Q4.7.1], page 56).

Q5.0.20: Is there a way to start a new XEmacs if there's no gnuserv running, and otherwise use gnuclient?

Jan Vroonhof writes:

> Here is one of the solutions, we have this in a script called '`etc/editclient.sh`'.
>
> ```
> #!/bin/sh
> if gnuclient -batch -eval t >/dev/null 2>&1
> then
>   exec gnuclient ${1+"$@"}
> else
>   xemacs -unmapped -f gnuserv-start &
>   until gnuclient -batch -eval t >/dev/null 2>&1
>   do
>      sleep 1
>   done
>   exec gnuclient ${1+"$@"}
> fi
> ```
>
> Note that there is a known problem when running XEmacs and 'gnuclient -nw' on the same TTY.

5.1: Emacs Lisp Programming Techniques

Q5.1.1: What is the difference in key sequences between XEmacs and GNU Emacs?

Erik Naggum writes;

> Emacs has a legacy of keyboards that produced characters with modifier bits, and therefore map a variety of input systems into this scheme even today. XEmacs is instead optimized for X events. This causes an incompatibility in the way key sequences are specified, but both Emacs and XEmacs will accept a key sequence as a vector of lists of modifiers that ends with a key, e.g., to bind M-C-a, you would say `[(meta control a)]` in both Emacsen. XEmacs has an abbreviated form for a single key, just (meta control a). Emacs has an abbreviated form for the Control and the Meta modifiers to string-characters (the ASCII characters), as in '`\M-\C-a`'. XEmacs users need to be aware that the abbreviated form works only for one-character key sequences, while Emacs users need to be aware that the string-character is rather limited. Specifically, the string-character can accommodate only 256 different values, 128 of which have the Meta modifier and 128 of which have not. In each of these blocks, only 32 characters have the Control modifier. Whereas `[(meta control A)]` differs from `[(meta control a)]` because the case differs, '`\M-\C-a`' and '`\M-\C-A`' do not. Programmers are advised to use the full common form, both because it is more readable and less error-prone, and because it is supported by both Emacsen.

Another (even safer) way to be sure of the key-sequences is to use the `read-kbd-macro` function, which takes a string like '`C-c <up>`', and converts it to the internal key representation of the Emacs you use. The function is available both on XEmacs and GNU Emacs.

Q5.1.2: Can I generate "fake" keyboard events?

I wonder if there is an interactive function that can generate fake keyboard events. This way, I could simply map them inside XEmacs.

This seems to work:

```
(defun cg--generate-char-event (ch)
  "Generate an event, as if ch has been typed"
  (dispatch-event (character-to-event ch)))

;;  Backspace and Delete stuff
(global-set-key [backspace]
  (lambda () (interactive) (cg--generate-char-event 127)))
(global-set-key [unknown_keysym_0x4]
  (lambda () (interactive) (cg--generate-char-event 4)))
```

Q5.1.3: Could you explain read-kbd-macro in more detail?

The `read-kbd-macro` function returns the internal Emacs representation of a human-readable string (which is its argument). Thus:

```
(read-kbd-macro "C-c C-a")
⇒ [(control ?c) (control ?a)]

(read-kbd-macro "C-c C-. <up>")
⇒ [(control ?c) (control ?.) up]
```

In GNU Emacs the same forms will be evaluated to what GNU Emacs understands internally—the sequences `"\C-x\C-c"` and `[3 67108910 up]`, respectively.

The exact human-readable syntax is defined in the docstring of `edmacro-mode`. I'll repeat it here, for completeness.

> Format of keyboard macros during editing:
>
> Text is divided into words separated by whitespace. Except for the words described below, the characters of each word go directly as characters of the macro. The whitespace that separates words is ignored. Whitespace in the macro must be written explicitly, as in foo SPC bar RET.
>
> - The special words RET, SPC, TAB, DEL, LFD, ESC, and NUL represent special control characters. The words must be written in uppercase.
> - A word in angle brackets, e.g., `<return>`, `<down>`, or `<f1>`, represents a function key. (Note that in the standard configuration, the function key `<return>` and the control key RET are synonymous.) You can use angle brackets on the words RET, SPC, etc., but they are not required there.
> - Keys can be written by their ascii code, using a backslash followed by up to six octal digits. This is the only way to represent keys with codes above \377.
> - One or more prefixes M- (meta), C- (control), S- (shift), A- (alt), H- (hyper), and s- (super) may precede a character or key notation. For function keys, the prefixes may go inside or outside of the brackets: `C-<down>` ≡ `<C-down>`. The prefixes may be written in any order: M-C-x ≡ C-M-x.

Prefixes are not allowed on multi-key words, e.g., C-abc, except that the Meta prefix is allowed on a sequence of digits and optional minus sign: M--123 ≡ M-- M-1 M-2 M-3

- The `^` notation for control characters also works: ^M ≡ C-m
- Double angle brackets enclose command names: `<<next-line>>` is shorthand for M-x next-line ⟨RET⟩.
- Finally, `REM` or `;;` causes the rest of the line to be ignored as a comment.

Any word may be prefixed by a multiplier in the form of a decimal number and `*`: `3*<right>` ≡ `<right> <right> <right>`, and `10*foo` ≡ `foofoofoofoofoofoofoofoofoofoo`.

Multiple text keys can normally be strung together to form a word, but you may need to add whitespace if the word would look like one of the above notations: `; ; ;` is a keyboard macro with three semicolons, but `;;;` is a comment. Likewise, `\ 1 2 3` is four keys but `\123` is a single key written in octal, and `< right >` is seven keys but `<right>` is a single function key. When in doubt, use whitespace.

Q5.1.4: What is the performance hit of let ?

In most cases, not noticeable. Besides, there's no avoiding `let`—you have to bind your local variables, after all. Some pose a question whether to nest `let`s, or use one `let` per function. I think because of clarity and maintenance (and possible future implementation), `let`-s should be used (nested) in a way to provide the clearest code.

Q5.1.5: What is the recommended use of setq ?

- Global variables

  You will typically `defvar` your global variable to a default value, and use `setq` to set it later.

  It is never a good practice to `setq` user variables (like `case-fold-search`, etc.), as it ignores the user's choice unconditionally. Note that `defvar` doesn't change the value of a variable if it was bound previously. If you wish to change a user-variable temporarily, use `let`:

  ```
  (let ((case-fold-search nil))
    ...                                    ; code with searches that must be case-s
    ...)
  ```

  You will notice the user-variables by their docstrings beginning with an asterisk (a convention).

- Local variables

  Bind them with `let`, which will unbind them (or restore their previous value, if they were bound) after exiting from the `let` form. Change the value of local variables with `setq` or whatever you like (e.g. `incf`, `setf` and such). The `let` form can even return one of its local variables.

  Typical usage:

  ```
  ;; iterate through the elements of the list returned by
  ;; 'hairy-function-that-returns-list'
  ```

```
(let ((l (hairy-function-that-returns-list)))
  (while l
    ... do something with (car l) ...
    (setq l (cdr l))))
```

Another typical usage includes building a value simply to work with it.

```
;; Build the mode keymap out of the key-translation-alist
(let ((inbox (file-truename (expand-file-name box)))
      (i 0))
  ... code dealing with inbox ...
  inbox)
```

This piece of code uses the local variable `inbox`, which becomes unbound (or regains old value) after exiting the form. The form also returns the value of `inbox`, which can be reused, for instance:

```
(setq foo-processed-inbox
      (let .....))
```

Q5.1.6: What is the typical misuse of `setq` ?

A typical misuse is probably `setq`ing a variable that was meant to be local. Such a variable will remain bound forever, never to be garbage-collected. For example, the code doing:

```
(defun my-function (whatever)
  (setq a nil)
  ... build a large list ...
  ... and exit ...)
```

does a bad thing, as `a` will keep consuming memory, never to be unbound. The correct thing is to do it like this:

```
(defun my-function (whatever)
  (let (a)                              ; default initialization is to nil
    ... build a large list ...
    ... and exit, unbinding ‘a’ in the process  ...)
```

Not only is this prettier syntactically, but it makes it possible for Emacs to garbage-collect the objects which `a` used to reference.

Note that even global variables should not be `setq`ed without `defvar`ing them first, because the byte-compiler issues warnings. The reason for the warning is the following:

```
(defun flurgoze nil)                    ; ok, global internal variable
...

(setq flurghoze t)                      ; ops!  a typo, but semantically correct.
                                        ; however, the byte-compiler warns.

While compiling toplevel forms:
** assignment to free variable flurghoze
```

Q5.1.7: I like the `do` form of cl, does it slow things down?

It shouldn’t. Here is what Dave Gillespie has to say about cl.el performance:

Many of the advanced features of this package, such as `defun*`, `loop`, and `setf`, are implemented as Lisp macros. In byte-compiled code, these complex notations will be expanded into equivalent Lisp code which is simple and efficient. For example, the forms

```
(incf i n)
(push x (car p))
```

are expanded at compile-time to the Lisp forms

```
(setq i (+ i n))
(setcar p (cons x (car p)))
```

which are the most efficient ways of doing these respective operations in Lisp. Thus, there is no performance penalty for using the more readable `incf` and `push` forms in your compiled code.

Interpreted code, on the other hand, must expand these macros every time they are executed. For this reason it is strongly recommended that code making heavy use of macros be compiled. (The features labelled Special Form instead of Function in this manual are macros.) A loop using `incf` a hundred times will execute considerably faster if compiled, and will also garbage-collect less because the macro expansion will not have to be generated, used, and thrown away a hundred times.

You can find out how a macro expands by using the `cl-prettyexpand` function.

Q5.1.8: I like recursion, does it slow things down?

Yes. Emacs byte-compiler cannot do much to optimize recursion. But think well whether this is a real concern in Emacs. Much of the Emacs slowness comes from internal mechanisms such as redisplay, or from the fact that it is an interpreter.

Please try not to make your code much uglier to gain a very small speed gain. It's not usually worth it.

Q5.1.9: How do I put a glyph as annotation in a buffer?

Here is a solution that will insert the glyph annotation at the beginning of buffer:

```
(make-annotation (make-glyph '([FORMAT :file FILE]
                               [string :data "fallback-text"]))
                 (point-min)
                 'text
                 (current-buffer))
```

Replace '`FORMAT`' with an unquoted symbol representing the format of the image (e.g. `xpm`, `xbm`, `gif`, `jpeg`, etc.) Instead of '`FILE`', use the image file name (e.g. '`/usr/local/lib/xemacs-21.4/etc/recycle.xpm`').

You can turn this to a function (that optionally prompts you for a file name), and inserts the glyph at `(point)` instead of `(point-min)`.

Q5.1.10: map-extents won't traverse all of my extents!

I tried to use `map-extents` to do an operation on all the extents in a region. However, it seems to quit after processing a random number of extents. Is it buggy?

No. The documentation of `map-extents` states that it will iterate across the extents as long as function returns `nil`. Unexperienced programmers often forget to return `nil` explicitly, which results in buggy code. For instance, the following code is supposed to delete all the extents in a buffer, and issue as many '`fubar!`' messages.

```
(map-extents (lambda (ext ignore)
               (delete-extent ext)
               (message "fubar!")))
```

Instead, it will delete only the first extent, and stop right there – because `message` will return a non-nil value. The correct code is:

```
(map-extents (lambda (ext ignore)
               (delete-extent ext)
               (message "fubar!")
               nil))
```

Q5.1.11: My elisp program is horribly slow. Is there

an easy way to find out where it spends time?

Hrvoje Niksic writes:

> Under XEmacs 20.4 and later you can use M-x profile-key-sequence , press a key (say ⟨RET⟩ in the Gnus Group buffer), and get the results using M-x profile-results . It should give you an idea of where the time is being spent.

Q5.2.1: How do I turn off the sound?

Add the following line to your '`init.el`'/'`.emacs`':

```
(setq bell-volume 0)
(setq sound-alist nil)
```

That will make your XEmacs totally silent—even the default ding sound (TTY beep on TTY-s) will be gone.

Starting with XEmacs 20.2 you can also change these with Customize. Select from the `Options` menu `Advanced (Customize)->Emacs->Environment->Sound->Sound...` or type M-x customize ⟨RET⟩ sound ⟨RET⟩.

Q5.2.2: How do I get funky sounds instead of a boring beep?

Make sure your XEmacs was compiled with sound support, and then put this in your '`init.el`'/'`.emacs`':

```
(load-default-sounds)
```

Q5.2.3: What's NAS, how do I get it?

See [Q2.0.3], page 13, for an explanation of the Network Audio System.

Q5.2.4: Sunsite sounds don't play.

I'm having some trouble with sounds I've downloaded from sunsite. They play when I run them through `showaudio` or cat them directly to '`/dev/audio`', but XEmacs refuses to play them.

Markus Gutschke writes:

> [Many of] These files have an (erroneous) 24byte header that tells about the format that they have been recorded in. If you cat them to '`/dev/audio`', the header will be ignored and the default behavior for /dev/audio will be used. This happens to be 8kHz uLaw. It is probably possible to fix the header by piping through `sox` and passing explicit parameters for specifying the sampling format; you then need to perform a 'null' conversion from SunAudio to SunAudio.

5.3: Miscellaneous

Q5.3.1: How do you make XEmacs indent CL if-clauses correctly?

I'd like XEmacs to indent all the clauses of a Common Lisp `if` the same amount instead of indenting the 3rd clause differently from the first two.

One way is to add, to '`init.el`'/'`.emacs`':

```
(put 'if 'lisp-indent-function nil)
```

However, note that the package `cl-indent` that comes with XEmacs sets up this kind of indentation by default. `cl-indent` also knows about many other CL-specific forms. To use `cl-indent`, one can do this:

```
(load "cl-indent")
(setq lisp-indent-function (function common-lisp-indent-function))
```

One can also customize '`cl-indent.el`' so it mimics the default `if` indentation `then` indented more than the `else`. Here's how:

```
(put 'if 'common-lisp-indent-function '(nil nil &body))
```

Also, a new version (1.2) of '`cl-indent.el`' was posted to comp.emacs.xemacs on 12/9/94. This version includes more documentation than previous versions. This may prove useful if you need to customize any indent-functions.

Q5.3.2: [This question intentionally left blank]

Obsolete question, left blank to avoid renumbering.

Q5.3.3: How can I print WYSIWYG a font-locked buffer?

Font-lock looks nice. How can I print (WYSIWYG) the highlighted document?

The package `ps-print`, which is now included with XEmacs, provides the ability to do this. The source code contains complete instructions on its use, in '`$prefix/lib/xemacs/xemacs-packages/lisp/ps-print/ps-print.el`', being the default location of an installed ps-print package.

Q5.3.4: Getting M-x lpr to work with postscript printer.

My printer is a Postscript printer and `lpr` only works for Postscript files, so how do I get M-x lpr-region and M-x lpr-buffer to work?

Put something like this in your '`init.el`'/'`.emacs`':

```
(setq lpr-command "a2ps")
(setq lpr-switches '("-p" "-1"))
```

If you don't use a2ps to convert ASCII to postscript (why not, it's free?), replace with the command you do use. Note also that some versions of a2ps require a '`-Pprinter`' to ensure spooling.

Q5.3.5: How do I specify the paths that XEmacs uses for finding files?

You can specify what paths to use by using a number of different flags when running configure. See the section MAKE VARIABLES in the top-level file INSTALL in the XEmacs distribution for a listing of those flags.

Most of the time, however, the simplest fix is: do not specify paths as you might for GNU Emacs. XEmacs can generally determine the necessary paths dynamically at run time. The only path that generally needs to be specified is the root directory to install into. That can be specified by passing the `--prefix` flag to configure. For a description of the XEmacs install tree, please consult the '`NEWS`' file.

Q5.3.6: [This question intentionally left blank]

Obsolete question, left blank to avoid renumbering.

Q5.3.7: Can I have the end of the buffer delimited in some way?

Say, with: '`[END]`'?

Try this:

```
(let ((ext (make-extent (point-min) (point-max))))
  (set-extent-property ext 'start-closed t)
  (set-extent-property ext 'end-closed t)
  (set-extent-property ext 'detachable nil)
  (set-extent-end-glyph ext (make-glyph [string :data "[END]"])))
```

Since this is XEmacs, you can specify an icon to be shown on window-system devices. To do so, change the `make-glyph` call to something like this:

```
(make-glyph '([xpm :file "~/something.xpm"]
              [string :data "[END]"]))
```

You can inline the xpm definition yourself by specifying `:data` instead of `:file`. Here is such a full-featured version that works on both X and TTY devices:

```
(let ((ext (make-extent (point-min) (point-max))))
  (set-extent-property ext 'start-closed t)
  (set-extent-property ext 'end-closed t)
  (set-extent-property ext 'detachable nil)
  (set-extent-end-glyph ext (make-glyph '([xpm :data "\
/* XPM */
static char* eye = {
\"20 11 7 2\",
\"__ c None\"
\"_` c #7f7f7f\",
\"_a c #fefefe\",
\"_b c #7f0000\",
\"_c c #fefe00\",
\"_d c #fe0000\",
\"_e c #bfbfbf\",
\"___________`_`_`___b_b_b_b_________`____\",
\"_________`_`_`___b_c_c_c_b_b____________\",
\"_____`_`_`_e___b_b_c_c_c___b___b_______`\",
\"___`_`_e_a___b_b_d___b___b___b___b______\",
\"_`_`_e_a_e___b_b_d_b___b___b___b___b____\",
\"_`_`_a_e_a___b_b_d___b___b___b___b___b__\",
\"_`_`_e_a_e___b_b_d_b___b___b___b___b_b__\",
\"___`_`_e_a___b_b_b_d_c___b___b___d_b____\",
\"_____`_`_e_e___b_b_b_d_c___b_b_d_b______\",
\"_`_____`_`_`_`___b_b_b_d_d_d_d_b________\",
\"___`_____`_`_`_`___b_b_b_b_b_b__________\",
} ;"]
                                          [string :data "[END]"]))))
```

Note that you might want to make this a function, and put it to a hook. We leave that as an exercise for the reader.

Q5.3.8: How do I insert today's date into a buffer?

Like this:

```
(insert (current-time-string))
```

Q5.3.9: Are only certain syntactic character classes available for abbrevs?

Markus Gutschke writes:

> Yes, abbrevs only expands word-syntax strings. While XEmacs does not prevent you from defining (e.g. with C-x a g or C-x a l) abbrevs that contain special characters, it will refuse to expand them. So you need to ensure, that the abbreviation contains letters and digits only. This means that ‘`xd`’, ‘`d5`’, and ‘`5d`’ are valid abbrevs, but ‘`&d`’, and ‘`x d`’ are not.
>
> If this sounds confusing to you, (re-)read the online documentation for abbrevs (C-h i m XEmacs RET m Abbrevs RET), and then come back and read this question/answer again.

Starting with XEmacs 20.3 this restriction has been lifted.

Q5.3.10: How can I get those oh-so-neat X-Face lines?

Firstly there is an ftp site which describes X-faces and has the associated tools mentioned below, at `ftp://ftp.cs.indiana.edu:/pub/faces/`.

Then the steps are

1. Create 48x48x1 bitmap with your favorite tool
2. Convert to "icon" format using one of xbm2ikon, pbmtoicon, etc., and then compile the face.
3.
   ```
   cat file.xbm | xbm2ikon |compface > file.face
   ```
4. Then be sure to quote things that are necessary for emacs strings:
   ```
   cat ./file.face | sed 's/\\/\\\\/g'

   \

   | sed 's/\"/\\\"/g' > ./file.face.quoted
   ```
5. Then set up emacs to include the file as a mail header - there were a couple of suggestions here—either something like:
   ```
   (setq  mail-default-headers
          "X-Face:  Ugly looking text string here")
   ```
   Or, alternatively, as:
   ```
   (defun mail-insert-x-face ()
     (save-excursion
       (goto-char (point-min))
       (search-forward mail-header-separator)
       (beginning-of-line)
       (insert "X-Face:")
       (insert-file-contents "~/.face")))

   (add-hook 'mail-setup-hook 'mail-insert-x-face)
   ```

However, 2 things might be wrong:

Some versions of pbmtoicon produces some header lines that is not expected by the version of compface that I grabbed. So I found I had to include a `tail +3` in the pipeline like this:

```
cat file.xbm | xbm2ikon | tail +3 |compface > file.face
```

Some people have also found that if one uses the `(insert-file)` method, one should NOT quote the face string using the sed script .

It might also be helpful to use Stig's script (included in the compface distribution at XEmacs.org) to do the conversion.

Contributors for this item:

Paul Emsley, Ricardo Marek, Amir J. Katz, Glen McCort, Heinz Uphoff, Peter Arius, Paul Harrison, and Vegard Vesterheim

Q5.3.11: How do I add new Info directories?

You use something like:

```
(setq Info-directory-list (cons
                           (expand-file-name "~/info")
                           Info-default-directory-list))
```

David Masterson writes:

> Emacs Info and XEmacs Info do many things differently. If you're trying to support a number of versions of Emacs, here are some notes to remember:
>
> 1. Emacs Info scans `Info-directory-list` from right-to-left while XEmacs Info reads it from left-to-right, so append to the *correct* end of the list.
> 2. Use `Info-default-directory-list` to initialize `Info-directory-list` *if* it is available at startup, but not all Emacsen define it.
> 3. Emacs Info looks for a standard '`dir`' file in each of the directories scanned from #1 and magically concatenates them together.
> 4. XEmacs Info looks for a '`localdir`' file (which consists of just the menu entries from a '`dir`' file) in each of the directories scanned from #1 (except the first), does a simple concatenation of them, and magically attaches the resulting list to the end of the menu in the '`dir`' file in the first directory.
>
> Another alternative is to convert the documentation to HTML with texi2html and read it from a web browser like Lynx or W3.

Q5.3.12: What do I need to change to make printing work?

For regular printing there are two variables that can be customized.

`lpr-command`

This should be set to a command that takes standard input and sends it to a printer. Something like:

```
(setq lpr-command "lp")
```

`lpr-switches`

This should be set to a list that contains whatever the print command requires to do its job. Something like:

```
(setq lpr-switches '("-depson"))
```

For postscript printing there are three analogous variables to customize.

`ps-lpr-command`

This should be set to a command that takes postscript on standard input and directs it to a postscript printer.

`ps-lpr-switches`

This should be set to a list of switches required for `ps-lpr-command` to do its job.

`ps-print-color-p`

This boolean variable should be set `t` if printing will be done in color, otherwise it should be set to `nil`.

NOTE: It is an undocumented limitation in XEmacs that postscript printing (the `Pretty Print Buffer` menu item) requires a window system environment. It cannot be used outside of X11.

6 XEmacs on MS Windows

This is part 6 of the XEmacs Frequently Asked Questions list, written by Hrvoje Niksic and others. This section is devoted to the MS Windows port of XEmacs.

6.0: General Info

Q6.0.1: What is the status of the XEmacs port to Windows?

Is XEmacs really ported to MS Windows? What is the status of the port?

Beginning with release 21.0, XEmacs has worked under MS Windows. A group of dedicated developers actively maintains and improves the Windows-specific portions of the code. The mailing list at `xemacs-nt@xemacs.org` is dedicated to that effort (please use the -request address to subscribe). (Despite its name, XEmacs actually works on all versions of Windows.)

As of May 2001, XEmacs on MS Windows is stable and full-featured, and has been so for a year or more – in fact, some features, such as printing, actually work better on Windows than native Unix. However, the internationalization (Mule) support does not work – although this is being actively worked on.

Q6.0.2: What flavors of MS Windows are supported? The list name implies NT only.

The list name is misleading, as XEmacs supports and has been compiled on Windows 95, Windows 98, Windows NT, Windows 2000, Windows ME, Windows XP, and all newer versions of Windows. The MS Windows-specific code is based on Microsoft Win32 API, and will not work on MS Windows 3.x or on MS-DOS.

XEmacs also supports the Cygwin and MinGW development and runtime environments, where it also uses native Windows code for graphical features.

Q6.0.3: Are binaries available?

Binaries are available at `http://www.xemacs.org/Download/win32/` for the native and Cygwin MS Windows versions of 21.4, and the native version of 21.1.

The 21.4 binaries use a modified version of the Cygwin installer. Run the provided ‘`setup.exe`’, and follow the instructions.

Q6.0.4: Can I build XEmacs on MS Windows with X support? Do I need to?

Yes, you can, but no you do not need to. In fact, we recommend that you use a native-GUI version unless you have a specific need for an X version.

Q6.0.5: I'd like to help out. What do I do?

It depends on the knowledge and time you possess. If you are a programmer, try to build XEmacs and see if you can improve it. Windows-specific improvements like integration with established Windows environments are especially sought after.

Otherwise, you can still help by downloading the binaries, using XEmacs as your everyday editor and reporting bugs you find to the mailing list.

Another area where we need help is the documentation: We need good documentation for building XEmacs and for using it. This FAQ is a small step in that direction.

Q6.0.6: What are Cygwin and MinGW, and do I need them to run XEmacs?

To answer the second part of the question: No, you, you don't need Cygwin or MinGW to build or to run XEmacs. But if you have them and want to use them, XEmacs supports these environments.

(One important reason to support Cygwin is that it lets the MS Windows developers test out their code in a Unix environment without actually having to have a Unix machine around. For this reason alone, Cygwin support is likely to remain supported for a long time in XEmacs. Same goes for the X support under Cygwin, for the same reasons. MinGW support, on the other hand, depends on volunteers to keep it up to date; but this is generally not hard.)

Cygwin is a set of tools providing Unix-like API on top of Win32. It makes it easy to port large Unix programs without significant changes to their source code. It is a development environment as well as a runtime environment.

When built with Cygwin, XEmacs supports all display types – TTY, X & Win32 GUI, and can be built with support for all three simultaneously. If you build with Win32 GUI support then the Cygwin version uses the majority of the Windows-specific code, which is mostly related to display. If you want to build with X support you need X libraries (and an X server to display XEmacs on); see [Q6.1.4], page 77. TTY and Win32 GUI require no additional libraries beyond what comes standard with Cygwin.

The advantages of the Cygwin version are that it integrates well with the Cygwin environment for existing Cygwin users; uses configure so building with different features is very easy; and actively supports X & TTY. Furthermore, the entire Cygwin environment and compiler are free, whereas Visual C++ costs money.

The disadvantage is that it requires the whole Cygwin environment, whereas the native port requires only a suitable MS Windows compiler. Also, it follows the Unix filesystem and process model very closely (some will undoubtedly view this as an advantage).

See `http://sources.redhat.com/cygwin/` for more information on Cygwin.

MinGW is a collection of header files and import libraries that allow one to use GCC under the Cygwin environment to compile and produce exactly the same native Win32 programs that you can using Visual C++. Programs compiled with MinGW make use of the standard Microsoft runtime library '`MSVCRT.DLL`', present on all Windows systems, and look, feel, and act like a standard Visual-C-produced application. (The only difference is the compiler.) This means that, unlike a standardly-compiled Cygwin application, no extra runtime support (e.g. Cygwin's '`cygwin1.dll`') is required. This, along with the fact that GCC is free (and works in a nice Unix-y way in a nice Unix-y environment, for those die-hard Unix hackers out there), is the main advantage of MinGW. It is also potentially faster than Cygwin because it has less overhead when calling Windows, but you lose the POSIX emulation layer, which makes Unix programs harder to port. (But this is irrelevant for XEmacs since it's already ported to Win32.)

See `http://www.mingw.org/` for more information on MinGW.

Q6.0.7: What exactly are all the different ways to build XEmacs under Windows?

XEmacs can be built in several ways in the MS Windows environment.

The standard way is what we call the "native" port. It uses the Win32 API and has no connection with X whatsoever – it does not require X libraries to build, nor does it require an X server to run. The native port is the most reliable version and provides the best graphical support. Almost all development is geared towards this version, and there is little reason not to use it.

The second way to build is the Cygwin port. It takes advantage of Cygnus emulation library under Win32. See [Q6.0.6], page 75, for more information.

A third way is the MinGW port. It uses the Cygwin environment to build but does not require it at runtime. See [Q6.0.6], page 75, for more information.

Finally, you might also be able to build the non-Cygwin, non-MinGW "X" port. This was actually the first version of XEmacs that ran under MS Windows, and although the code is still in XEmacs, it's essentially orphaned and it's unlikely it will compile without a lot of work. If you want an MS Windows versin of XEmacs that supports X, use the Cygwin version. (The X support there is actively maintained, so that Windows developers can test the X support in XEmacs.)

6.1: Building XEmacs on MS Windows

Q6.1.1: What compiler/libraries do I need to compile XEmacs?

You need Visual C++ 4.2, 5.0, or 6.0 for the native version. (We have some beta testers currently trying to compile with VC.NET, aka version 7.0, but we can't yet report complete success.) For the Cygwin and MinGW versions, you need the Cygwin environment, which comes with GCC, the compiler used for those versions. See [Q6.0.6], page 75, for more information on Cygwin and MinGW.

Q6.1.2: How do I compile the native port?

Please read the file '`nt/README`' in the XEmacs distribution, which contains the full description.

Q6.1.3: What do I need for Cygwin?

You can find the Cygwin tools and compiler at:

http://sources.redhat.com/cygwin/

Click on the ‘`Install now!`’ link, which will download a file ‘`setup.exe`’, which you can use to download everything else. (You will need to pick a mirror site; ‘`mirrors.rcn.net`’ is probably the best.) You should go ahead and install everything – you’ll get various ancillary libraries that XEmacs needs or likes, e.g. XPM, PNG, JPEG, TIFF, etc.

If you want to compile under X, you will also need the X libraries; see [Q6.1.6], page 78.

If you want to compile without X, you will need the ‘`xpm-nox`’ library, which must be specifically selected in the Cygwin netinstaller; it is not selected by default. The package has had various names. Currently it is called ‘`cygXpm-noX4.dll`’.

Q6.1.4: How do I compile under Cygwin?

Similar as on Unix; use the usual ‘configure’ and ‘make’ process. Some problems to watch out for:

- make sure HOME is set. This controls where you ‘`init.el`’/‘`.emacs`’ file comes from;
- CYGWIN needs to be set to tty for process support to work, e.g. CYGWIN=tty;
- picking up some other grep or other UNIX-like tools can kill configure;
- static heap too small, adjust ‘`src/sheap-adjust.h`’ to a more positive number;
- (Unconfirmed) The Cygwin version doesn’t understand ‘`//machine/path`’ type paths so you will need to manually mount a directory of this form under a unix style directory for a build to work on the directory;
- If you’re building WITHOUT X11, don’t forget to change symlinks ‘`/usr/lib/libXpm.a`’ and ‘`/usr/lib/libXpm.dll.a`’ to point to the non-X versions of these libraries. By default they point to the X versions. So:

  ```
  /usr/lib/libXpm.a     -> /usr/lib/libXpm-noX.a
  /usr/lib/libXpm.dll.a -> /usr/lib/libXpm-noX.dll.a
  ```

  (This advice may now be obsolete because of the availability of the cygXpm-noX4.dll package from Cygwin. Send confirmation to faq@xemacs.org.)
- Other problems are listed in the ‘`PROBLEMS`’ file, in the top-level directory of the XEmacs sources.

Q6.1.5: How do I compile using MinGW (aka ` the -mno-cygwin flag to gcc ')?

Similar to the method for Unix. Things to remember:

- Specify the target host on the command line for ‘`./configure`’, e.g. ‘`./configure i586-pc-mingw32`’.
- Be sure that your build directory is mounted such that it has the same path either as a cygwin path (‘`/build/xemacs`’) or as a Windows path (‘`c:\build\xemacs`’).
- Build ‘`gcc -mno-cygwin`’ versions of the extra libs, i.e. ‘`libpng`’, ‘`compface`’, etc.
- Specify the target location of the extra libs on the command line to ‘`configure`’, e.g. ‘`./configure --site-prefixes=/build/libs i586-pc-mingw32`’.

Q6.1.6: I decided to run with X. Where do I get an X server?

As of May 2001, we are recommending that you use the port of XFree86 to Cygwin. This has recently stabilized, and will undoubtedly soon make most other MS Windows X servers obsolete. It is what the Windows developers use to test the MS Windows X support.

To install, go to `http://xfree86.cygwin.com/`. There is a detailed description on that site of exactly how to install it. This installation also provides the libraries, include files, and other stuff needed for development; a large collection of internationalized fonts; the standard X utilities (xterm, twm, etc.) – in a word, the works.

NOTE: As of late May 2001, there is a bug in the file '`startxwin.bat`', used to start X Windows. It passes the option '`-engine -4`' to the X server, which is bogus – you need to edit the file and change it to '`-engine 4`'.

Q6.1.7: How do I compile with X support?

To compile under Cygwin, all you need to do is install XFree86 (see [Q6.1.6], page 78). Once installed, '`configure`' should automatically find the X libraries and compile with X support.

As noted above, the non-Cygwin X support is basically orphaned, and probably won't work. But if it want to try, it's described in '`nt/README`' in some detail. Basically, you need to get X11 libraries from ftp.x.org, and compile them. If the precompiled versions are available somewhere, we don't know of it.

6.2: Customization and User Interface

Q6.2.1: How does the port cope with differences in the Windows user interface?

The XEmacs (and Emacs in general) user interface is pretty different from what is expected of a typical MS Windows program. How does the MS Windows port cope with it?

As a general rule, we follow native MS Windows conventions as much as possible. 21.4 is a fairly complete Windows application, supporting native printing, system file dialog boxes, tool tips, etc. In cases where there's a clear UI conflict, we currently use normal Unix XEmacs behavior by default, but make sure the MS Windows "look and feel" (mark via shift-arrow, self-inserting deletes region, Alt selects menu items, etc.) is easily configurable (respectively: using the variable `shifted-motion-keys-select-region` in 21.4 and above [it's in fact the default in these versions], or the '`pc-select`' package; using the '`pending-del`' package; and setting the variable `menu-accelerator-enabled` to `menu-force` in 21.4 and above). In fact, if you use the sample '`init.el`' file as your init file, you will get all these behaviors automatically turned on.

In future versions, some of these features might be turned on by default in the MS Windows environment.

Q6.2.2: How do I change fonts in XEmacs on MS Windows?

In 21.4 and above, you can use the "Options" menu to change the font. You can also do it in your init file, e.g. like this:

```
(set-face-font 'default "Lucida Console:Regular:10")
(set-face-font 'modeline "MS Sans Serif:Regular:10")
```

Q6.2.3: Where do I put my ‘init.el’/‘.emacs’ file?

‘`init.el`’ is the name of the init file starting with 21.4, and is located in the subdirectory ‘`.xemacs/`’ of your home directory. In prior versions, the init file is called ‘`.emacs`’ and is located in your home directory. Your home directory under Windows is determined by the ‘`HOME`’ environment variable. If this is not set, it defaults to ‘`C:\`’.

To set this variable, modify ‘`AUTOEXEC.BAT`’ under Windows 95/98, or select ‘`Control Panel->System->Advanced->Environment Variables...`’ under Windows NT/2000.

Q6.2.4: How do I get Windows Explorer to associate a file type with XEmacs?

Associating a new file type with XEmacs.

In Explorer select ‘`View/Options/File Types`’, press ‘`[New Type...]`’ and fill in the dialog box, e.g.:

```
Description of type:    Emacs Lisp source
Associated extension:   el
Content Type (MIME):    text/plain
```

then press ‘`[New...]`’ and fill in the ‘`Action`’ dialog box as follows:

```
Action:
Open

Application used to perform action:
D:\Full\path\for\xemacs.exe "%1"

[x] Use DDE

DDE Message:
open("%1")

Application:
<leave blank>

DDE Application Not Running:
<leave blank>

Topic:
<leave blank>
```

Associating an existing file type with XEmacs.

In Explorer select ‘`View/Options/File Types`’. Click on the file type in the list and press ‘`[Edit...]`’. If the file type already has an ‘`Open`’ action, double click on it and fill in the ‘`Action`’ dialog box as described above; otherwise create a new action.

If the file type has more than one action listed, you probably want to make the ‘`Open`’ action that you just edited the default by clicking on it and pressing ‘`Set Default`’.

Note for Windows 2000 users: Under Windows 2000, get to ‘`File Types`’ using ‘`Control Panel->Folder Options->File Types`’.

Q6.2.5: Is it possible to print from XEmacs?

As of 21.4, printing works on Windows, using simply ‘`File->Print`’, and can be configured with ‘`File->Page Setup`’.

Prior to 21.4, there is no built-in support, but there are some clever hacks out there. If you know how, please let us know and we’ll put it here.

6.3: Miscellaneous

Q6.3.1: Does XEmacs rename all the `win32-*' symbols to `w32-*'?

In his flavor of Emacs 20, Richard Stallman has renamed all the ‘`win32-*`’ symbols to ‘`w32-*`’. Does XEmacs do the same?

We consider such a move counter-productive, thus we do not use the ‘`w32`’ prefix. (His rather questionable justification was that he did not consider Windows to be a "winning" platform.) However, the name ‘`Win32`’ is not particularly descriptive outside the Windows world, and using just ‘`windows-`’ would be too generic. So we chose a compromise, the prefix ‘`mswindows-`’ for Windows-related variables and functions.

Thus all the XEmacs variables and functions directly related to either the Windows GUI or OS are prefixed ‘`mswindows-`’ (except for a couple of debugging variables, prefixed ‘`debug-mswindows-`’). From an architectural perspective, however, we believe that this is mostly a non-issue because there should be a very small number of window-systems-specific variables anyway. Whenever possible, we try to provide generic interfaces that apply to all window systems.

Q6.3.2: What are the differences between the various MS Windows emacsen?

XEmacs, Win-Emacs, DOS Emacs, NT Emacs, this is all very confusing. Could you briefly explain the differences between them?

Here is a recount of various Emacs versions running on MS Windows:

- XEmacs
  - Beginning with XEmacs 19.12, XEmacs’ architecture was redesigned in such a way to allow clean support of multiple window systems. At this time the TTY support was added, making X and TTY the first two "window systems" supported by XEmacs. The 19.12 design is the basis for the current native MS Windows code.

  - Some time during 1997, David Hobley (soon joined by Marc Paquette) imported some of the NT-specific portions of GNU Emacs, making XEmacs with X support compile under Windows NT, and creating the "X" port.
  - Several months later, Jonathan Harris sent out initial patches to use the Win32 API, thus creating the native port. Since then, various people have contributed, including Kirill M. Katsnelson (contributed support for menubars, subprocesses and network, as well as loads of other code), Andy Piper (ported XEmacs to Cygwin environment, contributed Windows unexec, Windows-specific glyphs and toolbars code, and more), Ben Wing (loads of improvements; primary MS Windows developer since 2000), Jeff Sparkes (contributed scrollbars support) and many others.
- NT Emacs
  - NT Emacs is a version of GNU Emacs modified to compile and run under MS Windows 95 and NT using the native Win32 API. As such, it is close in spirit to the XEmacs "native" port.
  - NT Emacs has been written by Geoff Voelker, and more information can be found at
    `http://www.gnu.org/software/emacs/windows/ntemacs.html`.
- Win-Emacs
  - Win-Emacs was a port of Lucid Emacs 19.6 to MS Windows using X compatibility libraries. Win-Emacs was written by Ben Wing. The MS Windows code never made it back to Lucid Emacs, and its creator (Pearl Software) has long since gone out of business.
- GNU Emacs for DOS
  - GNU Emacs features support for MS-DOS and DJGPP (D.J. Delorie's DOS port of GCC). Such an Emacs is heavily underfeatured, because it does not support long file names, lacks proper subprocesses support, and is far too big compared with typical DOS editors.
- GNU Emacs compiled with Win32
  - Starting with version 19.30, it has been possible to compile GNU Emacs under MS Windows using the DJGPP compiler and X libraries. The result is very similar to GNU Emacs compiled under MS DOS, only it works somewhat better because it runs in 32-bit mode, makes use of all the system memory, supports long file names, etc.

Q6.3.3: XEmacs 21.1 on Windows used to spawn an ugly console window on every startup. Has that been fixed?

Yes.

The console was there because '`temacs`' (and in turn, '`xemacs`') was a console application, and Windows typically creates a new console for a console process unless the creating process requests that one isn't created. This used to be fixed with '`runemacs`', a small Windows application that existed merely to start '`xemacs`', stating that it didn't want a console.

XEmacs 21.4 fixes this cleanly by the virtue of being a true "GUI" application. The explanation of what that means is included for educational value.

When building an application to be run in a Win32 environment, you must state which sub-system it is to run in. Valid subsystems include "console" and "gui". The subsystem you use affects the run time libraries linked into your application, the start up function that is run before control is handed over to your application, the entry point to your program, and how Windows normally invokes your program. (Console programs automatically get a console created for them at startup if their stdin/stdout don't point anywhere useful, which is the case when run from the GUI. This is a stupid design, of course – instead, the console should get created only when the first I/O actually occurs! GUI programs have an equally stupid design: When called from '`CMD.EXE`'/'`COMMAND.COM`', their stdin/stdout will be set to point nowhere useful, even though the command shell has its own stdin/stdout. It's as if someone who had learned a bit about stdio but had no actual knowledge of interprocess communication designed the scheme; unfortunately, the whole process-communication aspect of the Win32 API is equally badly designed.) For example, the entry point for a console app is "main" (which is what you'd expect for a C/C++ program), but the entry point for a "gui" app is "WinMain". This confuses and annoys a lot of programmers who've grown up on Unix systems, where the kernel doesn't really care whether your application is a gui program or not.

For reasons not altogether clear, and are lost in the mists of time and tradition, XEmacs on Win32 started out as a console application, and therefore a console was automatically created for it. (It may have been made a console application partly because a console is needed in some circumstances, especially under Win95, to interrupt, terminate, or send signals to a child process, and because of the bogosity mentioned above with GUI programs and the standard command shell. Currently, XEmacs just creates and immediately hides a console when necessary, and works around the "no useful stdio" problem by creating its own console window as necessary to display messages in.)

Q6.3.4: What is the porting team doing at the moment?

(as of June 2001)

The porting team is continuing work on the MS Windows-specific code. Major projects are the development of Mule (internationalization) support for Windows and the improvement of the widget support (better support for dialog boxes, buttons, edit fields, and similar UI elements).

6.3: Troubleshooting

Q6.4.1 XEmacs won't start on Windows.

XEmacs relies on a process called "dumping" to generate a working executable. Under MS-Windows this process effectively fixes the memory addresses of information in the executable. When XEmacs starts up it tries to reserve these memory addresses so that the dumping process can be reversed – putting the information back at the correct addresses. Unfortunately some .DLLs (for instance the soundblaster driver) occupy memory addresses that can conflict with those needed by the dumped XEmacs executable. In this instance XEmacs will fail to start without any explanation. Note that this is extremely machine specific.

21.1.10 includes a fix for this that makes more intelligent guesses about which memory addresses will be free, and this should cure the problem for most people. 21.4 implements "portable dumping", which eliminates the problem altogether. We recommend you use the 21.4 binaries, but you can use the 21.1 binaries if you are very paranoid about stability. See [Q6.0.3], page 74.

Q6.4.2 Why do I get a blank toolbar on Windows 95?

You need at least version 4.71 of the system file ‘`comctl32.dll`’. The updated version is supplied with Internet Explorer 4 and later but if you are avoiding IE you can also download it from the Microsoft web site. Go into support and search for ‘`comctl32.dll`’. The download is a self-installing executable.

Q6.4.3 XEmacs complains "No such file or directory, diff"

or "ispell" or other commands that seem related to whatever you just tried to do (M-x ediff or M-$, for example).

There are a large number of common (in the sense that "everyone has these, they really do") Unix utilities that are not provided with XEmacs. The GNU Project's implementations are available for Windows in the the Cygwin distribution (`http://www.cygwin.com/`), which also provides a complete Unix emulation environment (and thus makes ports of Unix utilities nearly trivial). Another implementation is that from MinGW (`http://www.mingw.org/msys.shtml`). If you know of others, please let us know!

7 What the Future Holds

This is part 7 of the XEmacs Frequently Asked Questions list. This section will change frequently, and (in theory) should contain any interesting items that have transpired recently. (But in practice it's not getting updated like this.)

This section also contains descriptions of the new features in all the recent releases of XEmacs. For the most part, the information below is a synopsis of the more complete information that can be found in the file ‘`NEWS`’ in the ‘`etc`’ directory of the XEmacs distribution. You can view this file in XEmacs using C-h n or the ‘`Help`’ menu.

Information on older versions of XEmacs can be find in ‘`ONEWS`’ in the same directory, or ‘`OONEWS`’ for really old versions.

7.0: Changes

Q7.0.1: What new features will be in XEmacs soon?

Not yet written.

Q7.0.2: What's new in XEmacs 21.4?

21.4 was the "stable" version of the 21.2 series, which was considered "experimental" throughout its life; thus there were no "official" releases at all. In essence, XEmacs is now following the "alternating" scheme of Linux, where at any point there are at least two different development branches, one "stable" and one "experimental". Periodic releases happen in both branches, but those in the experimental branch are not tested as well, and there's no guarantee they will work at all. The experiemental branch is open to any and all code that's acceptable to the developers; the stable branch, however, is in general limited only to bug fixes, and all contributions are carefully reviewed to make sure they will increase and not decrease stability.

21.3 never existed at all; it was decided to follow the Linux scheme exactly, where odd-numbered series are experimental and even-numbered ones stable.

The following lists summarizes the essential changes made in this version. For a fuller list, see the '`NEWS`' in the '`etc`' directory of the XEmacs distribution, or use C-h n or the '`Help`' menu to view this file inside of XEmacs.

User-visible changes in XEmacs 21.4

- The delete key now deletes forward by default.
- Shifted motion keys now select text by default.
- You can now build XEmacs with support for GTK+ widget set.
- ~/.xemacs/init.el is now the preferred location for the init file. (XEmacs now supports a '~/.xemacs/init.el' startup file. Custom file will move to ~/.xemacs/custom.el.)
- Much-improved sample init.el, showing how to use many useful features.
- XEmacs support for menu accelerators has been much improved.
- Default menubar improvements. (Default menubar has many new commands and better organization. The font-menu is now available under MS Windows.)
- Dialog box improvements, including a real file dialog box. (XEmacs now has a proper file dialog box under MS Windows (and GTK)! The old clunky file dialog box is improved. Keyboard traversal now works correctly in MS Windows dialog boxes. There is a Search dialog box available from Edit->Find...)
- New buffer tabs.
- There is a new MS Windows installer, netinstall, ported from Cygwin.
- The subprocess quote-handling mechanism under Windows is much improved.
- Printing support now available under MS Windows.
- Selection improvements. (Kill and yank now interact with the clipboard under Windows. MS Windows support for selection is now much more robust. Motif selection support is now more correct (but slower).)
- Mail spool locking now works correctly.
- International support changes. (The default coding-priority-list is now safer. International keysyms are now supported under X. MS Windows 1251 code page now supported. Czech, Thai, Cyrillic-KOI8, Vietnamese, Ethiopic now supported. Proper support for words in Latin 3 and Latin 4.)

- Help buffers contain hyperlinks, and other changes.
- The modeline's text is now scrollable.
- The mouse wheel under MS Windows now functions correctly.
- Interactive searching and matching case improvements. (Incremental search will now highlight all visible matches. Interactive searches always respect uppercase characters.)
- Rectangle functions rewritten to avoid inserting extra spaces.
- New command 'kill-entire-line' that always kills the entire line.
- Default values correctly stored in minibuffer histories.
- You can now create "indirect buffers", like in GNU Emacs.
- Pixel-based scrolling has been implemented.
- Operation progress can be displayed using graphical widgets.
- User names following a tilde can now be completed at file name prompts.
- XEmacs can now play sound using Enlightenment Sound Daemon (ESD).
- X-Face support is now available under MS Windows.
- The PostgreSQL Relational Database Management System is now supported.
- Indentation no longer indents comments that begin at column zero.
- Face and variable settings can have comments in Customize.
- New locations for early package hierarchies.
- The 'auto-save' library has been greatly improved.
- New variable 'mswindows-alt-by-itself-activates-menu'.
- Other init-file-related changes. (Init file in your home directory may be called '.emacs.el'. New command-line switches -user-init-file and -user-init-directory.)
- Etags changes. See '`NEWS`' for full details.

Lisp and internal changes in XEmacs 21.4

Not yet written.

Q7.0.3: What's new in XEmacs 21.1?

21.1 was the "stable" version of "experimental" 21.0 series. See [Q7.0.2], page 84.

The following lists summarizes the essential changes made in this version. For a fuller list, see the '`NEWS`' in the '`etc`' directory of the XEmacs distribution, or use C-h n or the '`Help`' menu to view this file inside of XEmacs.

User-visible changes in XEmacs 21.1

- XEmacs is now supported under Microsoft Windows 95/98 and Windows NT operating systems. To discuss Windows-specific issues, subscribe to the mailing list at `xemacs-nt-request@xemacs.org`.
- XEmacs has been unbundled into constituent installable packages.

- Other notable changes The '`Options`' menu has been ported to Custom; XEmacs now is able to choose X visuals and use private colormaps; You can drag the vertical divider of "horizontally" (side-by-side) split windows.
- Building changes XEmacs can be built with support for 31-bit Lisp integers and 32-bit pointers (previously, it was 28-bit integers and pointers); XEmacs can be built with LDAP support; '`dir`' files can be removed in the Info subsystem, and will be regenerated on-the-fly.
- New packages '`imenu`', '`popper`', '`gdb-highlight`'
- Package changes Many changes to '`cc-mode`', '`gnus`', '`gnuclient`'. See '`NEWS`' for full details.
- New commands, variables and functions `center-to-window-line` (like `recenter` but doesn't force a redisplay); variable `user-full-name` (customize what your full name looks like in mail); M-x customize-changed-options (customize options whose default values changes because you upgraded your XEmacs); M-x add-log-convert (converts an old-style ChangeLog buffer to new-style); M-x zap-up-to-char (like `zap-to-char` but doesn't delete the char searched for); commands to store, retrieve and increment numbers in registers, useful for macros.
- Changes to commands, variables, and functions M-x query-replace and friends operate only on the region when it's active; `echo-keystrokes` can now be a floating-point number; M-. searches exact tag matches before inexact ones; function `user-full-name` with no arguments returns the var `user-full-name`; a prefix arg to M-: and C-h c inserts the result in the current buffer.
- Other changes Under X, new application class '`XEmacs`'; byte-compilation of user-specs now works.
- XEmacs/Mule (internationalization) changes: Mule support now works on TTY's; Egg/SJ3 input method now officially supported (Quail and Egg/Skk already available through LEIM since 20.3); localized Japanese menubars if XEmacs is built with the right support.

Lisp and internal changes in XEmacs 21.1

- Specifier changes The window locale now has a higher precedence than the buffer locale when instantiating; new macro `let-specifier`; new specifiers `vertical-scrollbar-visible-p`, horizontal-scrollbar-visible-p', `scrollbar-on-left-p`, `scrollbar-on-top-p`, `vertical-divider-always-visible-p`, `vertical-divider-shadow-thickness`, `vertical-divider-line-width`, `vertical-divider-spacing`; specifiers and symbols whose value is a specifier allowed as modeline specifications.
- Frame focus changes `focus-follows-mouse` works like FSF, prevents any attempt to permanently change the selected frame; new function `focus-frame` sets the window system focus a frame; new special forms `save-selected-frame` and `with-selected-frame`.
- Window function changes `select-window` now has optional argument NORECORD to inhibit recording a buffer change; `vertical-motion` now correctly handles optional WINDOW argument and has new optional argument PIXELS, to have the returned values be in pixels; new function `vertical-motion-pixels`; new functions

`window-text-area-pixel-{width,height,edges}`; new functions `shrink-window-pixels` and `enlarge-window-pixels`; new function `window-displayed-text-pixel-height`.

- Other function changes Arithmetic comparison functions `<`, `>`, `=`, `/=` now accept a variable number of arguments; hashtables now have a consistent read/print syntax; keyword symbols cannot be set to a value other than themselves; `concat` no longer accepts integer arguments; new function `string`, like `list`, `vector`, etc.; new function `temp-directory` (OS-independent way to get a temp directory); `load-average` has optional argument USE-FLOATS; `make-event` implemented completely; new function `function-interactive` (returns a function's interactive spec); new functions `lmessage`, `lwarn` (printf-like versions of `display-wessage`, `display-warning`); new keyword `:version` to `defcustom`.
- Performance: when the new GNU Malloc aka Doug Lea Malloc is available, it will be used (better performance on libc6 Linux systems); tracking line-numbers in modeline is now efficient; profiling records a call-count of all called functions, retrievable through `profile-call-count-results`.
- Startup and path searching: code to assemble paths at startup rewritten for new package system; new function `split-path` (splits by `path-separator`); `Info-default-directory-list` obsolete, use `Info-directory-list` instead; site-lisp is deprecated and no longer on the load-path by default.

Q7.0.4: What's new in XEmacs 20.4?

XEmacs 20.4 is a bugfix release with no user-visible changes.

Q7.0.5: What's new in XEmacs 20.3?

XEmacs 20.3 was released in November 1997. It contains many bugfixes, and a number of new features, including Autoconf 2 based configuration, additional support for Mule (Multi-language extensions to Emacs), many more customizations, multiple frames on TTY-s, support for multiple info directories, an enhanced gnuclient, improvements to regexp matching, increased MIME support, and many, many synches with GNU Emacs 20.

The XEmacs/Mule support has been only seriously tested in a Japanese locale, and no doubt many problems still remain. The support for ISO-Latin-1 and Japanese is fairly strong. MULE support comes at a price—about a 30% slowdown from 19.16. We're making progress on improving performance and XEmacs 20.3 compiled without Mule (which is the default) is definitely faster than XEmacs 19.16.

XEmacs 20.3 is the first non-beta v20 release, and will be the basis for all further development.

Q7.0.6: What's new in XEmacs 20.2?

The biggest changes in 20.2 include integration of EFS (the next generation of ange-ftp) and AUC Tex (the Emacs subsystem that includes a major mode for editing Tex and LaTeX, and a lot of other stuff). Many bugs from 20.0 have been fixed for this release. 20.2 also contains a new system for customizing XEmacs options, invoked via M-x customize.

XEmacs 20.2 is the development release (20.0 was beta), and is no longer considered unstable.

For older news, see the file ‘`ONEWS`’ in the ‘`etc`’ directory of the XEmacs distribution.

8 New information about old XEmacsen

This is part 8 of the XEmacs Frequently Asked Questions list. It will occasionally be updated to reflect new information about versions which are no longer being revised by the XEmacs Project. The primary purpose is advice on compatibility of older XEmacsen with new packages and updated versions of packages, but bug fixes (which will not be applied to released XEmacsen, but users can apply themselves) are also accepted.

Q8.0.1: Gnus 5.10 won't display smileys in XEmacs 21.1.

Eric Eide wrote:

> Previously I wrote:
>
> Eric> Summary: with Gnus 5.10.1 in XEmacs 21.1.14, I don’t see Eric> any smileys :-(.
>
> After a bit of sleuthing, I discovered the essence of the problem. For me, the form:
>
> ```
> (with-temp-buffer
>   (insert-file-contents "foo.xpm")
>   (buffer-string))
> ```
>
> returns the empty string. This is because something somewhere replaces the XPM data with a glyph — I haven’t figured out where this occurs.

Kyle Jones replies:

> Do this:
>
> ```
> (setq format-alist nil)
> ```
>
> The image-mode stuff is gone from format-alist in the 21.4 branch, praise be.