User's Guide to pcl-cvs - the Emacs Front-End to CVS

release 1.05

Per Cederqvist

last updated 31 May 1993

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DE-

FECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program 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.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

1 Installation

This section describes the installation of pcl-cvs, the GNU Emacs CVS front-end. You should install not only the elisp files themselves, but also the on-line documentation so that your users will know how to use it. You can create typeset documentation from the file `pcl-cvs.texinfo` as well as an on-line info file. The following steps are also described in the file `INSTALL` in the source directory.

1.1 Installation of the pcl-cvs program

1. Edit the file `Makefile` to reflect the situation at your site. The only things you have to change is the definition of `lispdir` and `infodir`. The elisp files will be copied to `lispdir`, and the info file to `infodir`.
2. Configure pcl-cvs.el

   There are a couple of paths that you have to check to make sure that they match you system. They appear early in the file pcl-cvs.el.

   NOTE: If your system is running emacs 18.57 or earlier you MUST uncomment the line that says:

   ```
   (setq delete-exited-processes nil)
   ```

   Setting `delete-exited-processes` to `nil` works around a bug in emacs that causes it to dump core. The bug was fixed in emacs 18.58.
3. Release 1.05 and later of pcl-cvs requires parts of the Elib library, version 0.07 or later. Elib is available via anonymous ftp from prep.ai.mit.edu in `pub/gnu/elib-0.07.tar.z`, and from a lot of other sites that mirrors prep. Get Elib, and install it, before proceeding.
4. Type `make install` in the source directory. This will byte-compile all `.el` files and copy both the `.el` and the `.elc` into the directory you specified in step 1.

   If you don't want to install the `.el` files but only the `.elc` files (the byte-compiled files), you can type "`make install_elc`" instead of "`make install`".

   If you only want to create the compiled elisp files, but don't want to install them, you can type `make elcfiles` instead. This is what happens if you only type `make` without parameters.
5. Edit the file `default.el` in your emacs lisp directory (usually `/usr/gnu/emacs/lisp` or something similar) and enter the contents of the file `pcl-cvs-startup.el` into it. It contains a couple of `auto-loads` that facilitates the use of pcl-cvs.

1.2 Installation of the on-line manual.

1. Create the info file 'pcl-cvs' from 'pcl-cvs.texinfo' by typing 'make info'. If you don't have the program 'makeinfo' you can get it by anonymous ftp from e.g. 'ftp.gnu.ai.mit.edu' as 'pub/gnu/texinfo-2.14.tar.Z' (there might be a newer version there when you read this), or you could use the preformatted info file 'pcl-cvs.info' that is included in the distribution (type 'cp pcl-cvs.info pcl-cvs').
2. Move the info file 'pcl-cvs' to your standard info directory. This might be called something like '/usr/gnu/emacs/info'.
3. Edit the file 'dir' in the info directory and enter one line to contain a pointer to the info file 'pcl-cvs'. The line can, for instance, look like this:

```
* Pcl-cvs: (pcl-cvs).       An Emacs front-end to CVS.
```

1.3 How to make typeset documentation from pcl-cvs.texinfo

If you have TEX installed at your site, you can make a typeset manual from 'pcl-cvs.texinfo'.

1. Run TEX by typing "make pcl-cvs.dvi". You will not get the indices unless you have the texindex program.
2. Convert the resulting device independent file 'pcl-cvs.dvi' to a form which your printer can output and print it. If you have a postscript printer there is a program, dvi2ps, which does. There is also a program which comes together with TEX, dvips, which you can use.

2 About pcl-cvs

Pcl-cvs is a front-end to CVS version 1.3. It integrates the most frequently used CVS commands into emacs.

2.1 Contributors to pcl-cvs

Contributions to the package are welcome. I have limited time to work on this project, but I will gladly add any code that you contribute to me to this package (see Chapter 8 [Bugs], page 26).

The following persons have made contributions to pcl-cvs.

- Brian Berliner wrote CVS, together with some other contributors. Without his work on CVS this package would be useless...
- Per Cederqvist wrote most of the otherwise unattributed functions in pcl-cvs as well as all documentation.
- Inge Wallin (`inge@lysator.liu.se`) wrote the skeleton to `pcl-cvs.texinfo`, and gave useful comments on it. He also wrote the files `elib-node.el` and `compile-all.el`. The file `cookie.el` was inspired by Inge.
- Linus Tolke (`linus@lysator.liu.se`) contributed useful comments on both the functionality and the documentation.
- Jamie Zawinski (`jwz@lucid.com`) contributed `pcl-cvs-lucid.el`.
- Leif Lonnblad contributed RCVS support.

Apart from these, a lot of people have send me suggestions, ideas, requests, bug reports and encouragement. Thanks a lot! Without your there would be no new releases of pcl-cvs.

2.2 Where can I get pcl-cvs?

The latest release of pcl-cvs can be fetched via anonymous ftp from `ftp.lysator.liu.se`, (IP no. 130.236.254.1) in the directory `pub/emacs`. If you don't live in Scandinavia you should probably check with archie to see if there is a site closer to you that archives pcl-cvs.

New releases will be announced to appropriate newsgroups. If you send your email address to me I will add you to my list of people to mail when I make a new release.

3 Getting started

This document assumes that you know what CVS is, and that you at least knows the fundamental concepts of CVS. If that is not the case you should read the man page for CVS.

Pcl-cvs is only useful once you have checked out a module. So before you invoke it you must have a copy of a module somewhere in the file system.

You invoke pcl-cvs by typing `M-x cvs-update RET`. If your emacs responds with '`[No match]`' your system administrator has not installed pcl-cvs properly. Try `M-x load-library RET pcl-cvs RET`. If that also fails - talk to your root. If it succeeds you might put this line in your '`.emacs`' file so that you don't have to type the '`load-library`' command every time you wish to use pcl-cvs:

```
(autoload 'cvs-update "pcl-cvs" nil t)
```

The function `cvs-update` will ask for a directory. The command '`cvs update`' will be run in that directory. (It should contain files that have been checked out from a CVS archive.) The output from `cvs` will be parsed and presented in a table in a buffer called '`*cvs*`'. It might look something like this:

```
PCL-CVS release 1.05.

In directory /users/ceder/FOO/test:
  Updated     bar
  Updated     file.txt
  Modified ci namechange
  Updated     newer

In directory /users/ceder/FOO/test/sub:
  Modified ci ChangeLog
---------- End -----
```

In this example the three files ('`bar`', '`file.txt`' and '`newer`') that are marked with '`Updated`' have been copied from the CVS repository to '`/users/ceder/FOO/test/`' since someone else have checked in newer versions of them. Two files ('`namechange`' and '`sub/ChangeLog`') have been modified locally, and needs to be checked in.

You can move the cursor up and down in the buffer with `C-n` and `C-p` or `n` and `p`. If you press `c` on one of the '`Modified`' files that file will be checked in to the CVS repository. See Section 5.4

[Committing changes], page 17. You can press `x` to get rid of the "uninteresting" files that have only been '`Updated`' (and don't require any further action from you).

You can also easily get a '`diff`' between your modified file and the base version that you started from, and you can get the output from '`cvs log`' and '`cvs status`' on the listed files simply by pressing a key (see Section 5.6 [Getting info about files], page 18).

4 Buffer contents

The display contains four columns. They contain, from left to right:

- An asterisk when the file is *marked* (see Section 4.2 [Selected files], page 15).
- The status of the file. See See Section 4.1 [File status], page 14, for more information.
- A "need to be checked in"-marker (`ci`).
- The file name.

4.1 File status

The `file status` field can have the following values:

`Updated` The file was brought up to date with respect to the repository. This is done for any file that exists in the repository but not in your source, and for files that you haven't changed but are not the most recent versions available in the repository.

`Modified`
The file is modified in your working directory, and there was no modification to the same file in the repository.

`Merged` The file is modified in your working directory, and there were modifications in the repository as well as in your copy, but they were merged successfully, without conflict, in your working directory.

`Conflict`
A conflict was detected while trying to merge your changes to *file* with changes from the source repository. *file* (the copy in your working directory) is now the output of the `rcsmerge` command on the two versions; an unmodified copy of your file is also in your working directory, with the name `.#`*file*`.`*version*, where *version* is the RCS revision that your modified file started from. See Section 5.11 [Viewing differences], page 20, for more details.

`Added` The file has been added by you, but it still needs to be checked in to the repository.

`Removed` The file has been removed by you, but it needs to be checked in to the repository. You can resurrect it by typing `a` (see Section 5.7 [Adding and removing files], page 18).

`Unknown` A file that was detected in your directory, but that neither appears in the repository, nor is present on the list of files that CVS should ignore.

There are also a few special cases, that rarely occur, which have longer strings in the fields:

'`Removed from repository`'
: The file has been removed from your directory since someone has removed it from the repository. (It is still present in the Attic directory, so no permanent loss has occurred). This, unlike the other entries in this table, is not an error condition.

'`Removed from repository, changed by you`'
: You have modified a file that someone have removed from the repository. You can correct this situation by removing the file manually (see see Section 5.7 [Adding and removing files], page 18).

'`Removed by you, changed in repository`'
: You have removed a file, and before you committed the removal someone committed a change to that file. You could use `a` to resurrect the file (see see Section 5.7 [Adding and removing files], page 18).

'`Move away` *file* `- it is in the way`'
: For some reason CVS does not like the file *file*. Rename or remove it.

'`This repository is missing! Remove this dir manually.`'
: It is impossible to remove a directory in the CVS repository in a clean way. Someone have tried to remove one, and CVS gets confused. Remove your copy of the directory.

4.2 Selected files

Many of the commands works on the current set of *selected* files.

- If there are any files that are marked they constitute the set of selected files.
- Otherwise, if the cursor points to a file, that file is the selected file.
- Otherwise, if the cursor points to a directory, all the files in that directory that appears in the buffer are the selected files.

This scheme might seem a little complicated, but once one get used to it, it is quite powerful.

See Section 5.3 [Marking files], page 17 tells how you mark and unmark files.

5 Commands

This chapter describes all the commands that you can use in pcl-cvs.

5.1 Updating the directory

`M-x cvs-update`
: Run a '`cvs update`' command. You will be asked for the directory in which the '`cvs update`' will be run. The output will be parsed by pcl-cvs, and the result printed in the '`*cvs*`' buffer (see see Chapter 4 [Buffer contents], page 14 for a description of the contents).

  By default, '`cvs-update`' will descend recursively into subdirectories. You can avoid that behavior by giving a prefix argument to it (e.g., by typing `C-u M-x cvs-update RET`).

  All other commands in pcl-cvs requires that you have a '`*cvs*`' buffer. This is the command that you use to get one.

  CVS uses lock files in the repository to ensure the integrity of the data files in the repository. They might be left behind i.e. if a workstation crashes in the middle of a CVS operation. CVS outputs a message when it is waiting for a lock file to go away. Pcl-cvs will show the same message in the *cvs* buffer, together with instructions for deleting the lock files. You should normally not have to delete them manually — just wait a little while and the problem should fix itself. But if the lock files doesn't disappear you can delete them with `M-x cvs-delete-lock RET`.

`g`
: This will run '`cvs update`' again. It will always use the same buffer that was used with the previous '`cvs update`'. Give a prefix argument to avoid descending into subdirectories. This runs the command '`cvs-mode-update-no-prompt`'.

5.2 Movement Commands

You can use most normal Emacs commands to move forward and backward in the buffer. Some keys are rebound to functions that take advantage of the fact that the buffer is a pcl-cvs buffer:

`SPC`
`C-n`

n These keys move the cursor one file forward, towards the end of the buffer (`cookie-next-cookie`).

C-p
p These keys move one file backward, towards the beginning of the buffer (`cookie-previous-cookie`).

5.3 Marking files

Pcl-cvs works on a set of *selected files* (see Section 4.2 [Selected files], page 15). You can mark and unmark files with these commands:

m This marks the file that the cursor is positioned on. If the cursor is positioned on a directory all files in that directory will be marked. (`cvs-mode-mark`).

u Unmark the file that the cursor is positioned on. If the cursor is on a directory, all files in that directory will be unmarked. (`cvs-mode-unmark`).

M Mark *all* files in the buffer (`cvs-mode-mark-all-files`).

ESC DEL Unmark *all* files (`cvs-mode-unmark-all-files`).

DEL Unmark the file on the previous line, and move point to that line (`cvs-mode-unmark-up`).

5.4 Committing changes

c All files that have a "need to be checked in"-marker (see Chapter 4 [Buffer contents], page 14) can be checked in with the `c` command. It checks in all selected files (see Section 4.2 [Selected files], page 15) (except those who lack the "ci"-marker - they are ignored). Pressing `c` causes `cvs-mode-commit` to be run.

When you press `c` you will get a buffer called '`*cvs-commit-message*`'. Enter the log message for the file(s) in it. When you are ready you should press `C-c C-c` to actually commit the files (using `cvs-edit-done`).

Normally the '`*cvs-commit-message*`' buffer will retain the log message from the previous commit, but if the variable `cvs-erase-input-buffer` is set to a non-`nil` value the buffer will be erased. Point and mark will always be located around the entire buffer so that you can easily erase it with `C-w` ('`kill-region`').

If you are editing the files in your emacs an automatic '`revert-buffer`' will be performed. (If the file contains '`$Id$`' keywords '`cvs commit`' will write a new file with the new values substituted. The auto-revert makes sure that you get them into your buffer). The revert will not occur if you have modified your buffer, or if '`cvs-auto-revert-after-commit`' is set to '`nil`'.

5.5 Editing files

There are currently three commands that can be used to find a file (that is, load it into a buffer and start editing it there). These commands work on the line that the cursor is situated at. They ignore any marked files.

`f` Find the file that the cursor points to. Run '`dired`' if the cursor points to a directory (`cvs-mode-find-file`).

`o` Like `f`, but use another window (`cvs-mode-find-file-other-window`).

`A` Invoke '`add-change-log-entry-other-window`' to edit a '`ChangeLog`' file. The '`ChangeLog`' will be found in the directory of the file the cursor points to. (`cvs-mode-add-change-log-entry-other-window`).

5.6 Getting info about files

Both of the following commands can be customized. See Chapter 6 [Customization], page 22.

`l` Run '`cvs log`' on all selected files, and show the result in a temporary buffer (`cvs-mode-log`).

`s` Run '`cvs status`' on all selected files, and show the result in a temporary buffer (`cvs-mode-status`).

5.7 Adding and removing files

The following commands are available to make it easy to add and remove files from the CVS repository.

`a` Add all selected files. This command can be used on '`Unknown`' files (see see Section 4.1 [File status], page 14). The status of the file will change to '`Added`', and you will have to use c ('cvs-mode-commit', see see Section 5.4 [Committing changes], page 17) to really add the file to the repository.

This command can also be used on 'Removed' files (before you commit them) to resurrect them.

Selected files that are neither 'Unknown' nor 'Removed' will be ignored by this command.

The command that is run is `cvs-mode-add`.

`r` This command removes the selected files (after prompting for confirmation). The files are 'rm'ed from your directory and (unless the status was 'Unknown'; see Section 4.1 [File status], page 14) they will also be 'cvs remove'd. If the files were 'Unknown' they will disappear from the buffer. Otherwise their status will change to 'Removed', and you must use c ('cvs-mode-commit', see Section 5.4 [Committing changes], page 17) to commit the removal.

The command that is run is `cvs-mode-remove-file`.

5.8 Undoing changes

`U` If you have modified a file, and for some reason decide that you don't want to keep the changes, you can undo them with this command. It works by removing your working copy of the file and then getting the latest version from the repository (`cvs-mode-undo-local-changes`.

5.9 Removing handled entries

`x` This command allows you to remove all entries that you have processed. More specifically, the lines for 'Updated' files (see Section 4.1 [File status], page 14 and files that have been checked in (see Section 5.4 [Committing changes], page 17) are removed from the buffer. If a directory becomes empty the heading for that directory is also removed. This makes it easier to get an overview of what needs to be done.

The command is called `cvs-mode-remove-handled`. If 'cvs-auto-remove-handled' is set to non-`nil` this will automatically be performed after every commit.

`C-k` This command can be used for lines that 'cvs-mode-remove-handled' would not delete, but that you want to delete (`cvs-mode-acknowledge`).

5.10 Ignoring files

`i` Arrange so that CVS will ignore the selected files. The file names are added to the '`.cvsignore`' file in the corresponding directory. If the '`.cvsignore`' doesn't exist it will be created.

The '`.cvsignore`' file should normally be added to the repository, but you could ignore it also if you like it better that way.

This runs `cvs-mode-ignore`.

5.11 Viewing differences

`d` Display a '`cvs diff`' between the selected files and the RCS version that they are based on. See Chapter 6 [Customization], page 22 describes how you can send flags to '`cvs diff`'. If *cvs-diff-ignore-marks* is set to a non-`nil` value or if a prefix argument is given (but not both) any marked files will not be considered to be selected. (`cvs-mode-diff-cvs`).

`b` If CVS finds a conflict while merging two versions of a file (during a '`cvs update`', see Section 5.1 [Updating the directory], page 16) it will save the original file in a file called '`.#`*FILE*`.`*VERSION*' where *FILE* is the name of the file, and *VERSION* is the RCS version number that your file was based on.

With the `b` command you can run a '`diff`' on the files '`.#`*FILE*`.`*VERSION*' and '*FILE*'. You can get a context- or Unidiff by setting '`cvs-diff-flags`' - see Chapter 6 [Customization], page 22. This command only works on files that have status '`Conflict`' or '`Merged`'.

If *cvs-diff-ignore-marks* is set to a non-`nil` value or if a prefix argument is given (but not both) any marked files will not be considered to be selected. (`cvs-mode-diff-backup`).

5.12 Running emerge

`e` Invoke '`emerge`' on one file. This command works slightly different depending on the file status.

'`Modified`'

Run '`emerge-files`' with your working file as file A, and the latest revision in the repository as file B.

'Merged'
'Conflict'
: Run 'emerge-files-with-ancestor' with your working file (as it was prior to your invocation of 'cvs-update') as file A, the latest revision in the repository as file B, and the revision that you based your local modifications on as ancestor.

Note: CVS has already performed a merge. The resulting file is not used in any way if you use this command. If you use the `q` command inside 'emerge' (to successfully terminate the merge) the file that CVS created will be overwritten.

5.13 Reverting your buffers

`R` If you are editing (or just viewing) a file in a buffer, and that file is changed by CVS during a 'cvs-update', all you have to do is type `R` in the *cvs* buffer to read in the new versions of the files.

All files that are 'Updated', 'Merged' or in 'Conflict' are reverted from the disk. Any other files are ignored. Only files that you were already editing are read.

An error is signalled if you have modified the buffer since it was last changed. (`cvs-mode-revert-updated-buffers`).

5.14 Miscellaneous commands

`M-x cvs-byte-compile-files`
: Byte compile all selected files that end in .el.

`M-x cvs-delete-lock`
: This command can be used in any buffer, and deletes the lock files that the *cvs* buffer informs you about. You should normally never have to use this command since CVS tries very carefully to always remove the lock files itself.

You can only use this command when a message in the *cvs* buffer tells you so. You should wait a while before using this command in case someone else is running a cvs command.

`q` Bury the *cvs* buffer. (`bury-buffer`).

6 Customization

If you have an idea about any customization that would be handy but isn't present in this list, please tell me! See Chapter 8 [Bugs], page 26 for info on how to reach me.

'cvs-erase-input-buffer'
: If set to anything else than nil the edit buffer will be erased before you write the log message (see Section 5.4 [Committing changes], page 17).

'cvs-inhibit-copyright-message'
: The copyright message that is displayed on startup can be annoying after a while. Set this variable to 't' if you want to get rid of it. (But don't set this to 't' in the system defaults file - new users should see this message at least once).

'cvs-diff-flags'
: A list of strings to pass as arguments to the 'cvs diff' and 'diff' programs. This is used by 'cvs-mode-diff-cvs' and 'cvs-mode-diff-backup' (key b, see Section 5.11 [Viewing differences], page 20). If you prefer the Unidiff format you could add this line to your '.emacs' file:

```
(setq cvs-diff-flags '("-u"))
```

'cvs-diff-ignore-marks'
: If this variable is non-nil or if a prefix argument is given (but not both) to 'cvs-mode-diff-cvs' or 'cvs-mode-diff-backup' marked files are not considered selected.

'cvs-log-flags'
: List of strings to send to 'cvs log'. Used by 'cvs-mode-log' (key l, see Section 5.6 [Getting info about files], page 18).

'cvs-status-flags'
: List of strings to send to 'cvs status'. Used by 'cvs-mode-status' (key s, see Section 5.6 [Getting info about files], page 18).

'cvs-auto-remove-handled'
: If this variable is set to any non-nil value 'cvs-mode-remove-handled' will be called every time you check in files, after the check-in is ready. See Section 5.9 [Removing handled entries], page 19.

'cvs-auto-revert-after-commit'
: If this variable is set to any non-'nil' value any buffers you have that visit a file that is committed will be automatically reverted. This variable is default 't'. See Section 5.4 [Committing changes], page 17.

'cvs-update-prog-output-skip-regexp'
: The '-u' flag in the 'modules' file can be used to run a command whenever a 'cvs update' is performed (see cvs(5)). This regexp is used to search for the last line in that output. It is normally set to '"$"'. That setting is only correct if the command outputs nothing. Note that pcl-cvs will get very confused if the command outputs *anything* to 'stderr'.

'cvs-cvsroot'
: This variable can be set to override 'CVSROOT'. It should be a string. If it is set then everytime a cvs command is run it will be called as 'cvs -d *cvs-cvsroot*...' This can be useful if your site has several repositories.

'TMPDIR'
: Pcl-cvs uses this *environment variable* to decide where to put the temporary files it needs. It defaults to '/tmp' if it is not set.

'cvs-commit-buffer-require-final-newline'
: When you enter a log message in the '*cvs-commit-message*' buffer pcl-cvs will normally automatically insert a trailing newline, unless there already is one. This behavior can be controlled via 'cvs-commit-buffer-require-final-newline'. If it is 't' (the default behavior), a newline will always be appended. If it is 'nil', newlines will never be appended. Any other value causes pcl-cvs to ask the user whenever there is no trailing newline in the commit message buffer.

'cvs-sort-ignore-file'
: If this variable is set to any non-'nil' value the '.cvsignore' will always be sorted whenever you use 'cvs-mode-ignore' to add a file to it. This option is on by default.

7 Future enhancements

Pcl-cvs is still under development and needs a number of enhancements to be called complete. Below is my current wish-list for future releases of pcl-cvs. Please, let me know which of these features you want most. They are listed below in approximately the order that I currently think I will implement them in.

- Rewritten parser code. There are many situations where pcl-cvs will fail to recognize the output from CVS. The situation could be greatly increased.
- '`cvs-status`'. This will run '`cvs status`' in a directory and produce a buffer that looks pretty much like the current *cvs* buffer. That buffer will include information for all version-controlled files. (There will be a simple keystroke to remove all "uninteresting" files, that is, files that are "Up-to-date"). In this new buffer you will be able to update a file, commit a file, et c. The big win with this is that you will be able to watch the differences between your current working file and the head revision in the repository before you update the file, and you can then choose to update it or let it wait for a while longer.
- Log mode. When this mode is finished you will be able to move around (using `n` and `p`) between the revisions of a file, mark two of them, and run a diff between them. You will be able to hide branches (similar to the way you can hide sub-paragraphs in outline-mode) and do merges between revisions. Other ideas about this are welcome.
- The current model for marks in the *cvs* buffer seems to be confusing. I am considering to use the VM model instead, where marks are normally inactive. To activate the mark, you issue a command like '`cvs-mode-next-command-uses-marks`'. I might implement a flag so that you can use either version. Feedback on this before I start coding it is very welcome.
- It should be possible to run commands such as '`cvs log`', '`cvs status`' and '`cvs commit`' directly from a buffer containing a file, instead of having to '`cvs-update`'. If the directory contains many files the '`cvs-update`' can take quite some time, especially on a slow machine. I planed to put these kind of commands on the prefix `C-c C-v`, but that turned out to be used by for instance `c++`-mode. If you have any suggestions for a better prefix key, please let me know.
- Increased robustness. For instance, you can not currently press `C-g` when you are entering the description of a file that you are adding without confusing pcl-cvs.
- Support for multiple active *cvs* buffers.
- Dired support. I have an experimental '`dired-cvs.el`' that works together with CVS 1.2. Unfortunately I wrote it on top of a non-standard '`dired.el`', so it must be rewritten.
- An ability to send user-supplied options to all the cvs commands.
- Pcl-cvs is not at all clever about what it should do when '`cvs update`' runs a program (due to the '`-u`' option in the '`modules`' file — see '`cvs(5)`'). The current release uses a regexp to

search for the end. At the very least that regexp should be configured for different modules. Tell me if you have any idea about what is the right thing to do. In a perfect world the program should also be allowed to print to 'stderr' without causing pcl-cvs to crash.

If you miss something in this wish-list, let me know! I don't promise that I will write it, but I will at least try to coordinate the efforts of making a good Emacs front end to CVS. See See Chapter 8 [Bugs], page 26 for information about how to reach me.

So far, I have written most of pcl-cvs in my all-to-rare spare time. If you want pcl-cvs to be developed faster you can write a contract with Signum Support to do the extension. You can reach Signum Support by email to 'info@signum.se' or via mail to Signum Support AB, Box 2044, S-580 02 Linkoping, Sweden. Phone: +46 (0) 13 - 21 46 00. Fax: +46 (0) 13 - 21 47 00.

8 Bugs (known and unknown)

If you find a bug or misfeature, don't hesitate to tell me! Send email to '`ceder@lysator.liu.se`'.

If you have ideas for improvements, or if you have written some extensions to this package, I would like to hear from you. I hope that you find this package useful!

Below is a partial list of currently known problems with pcl-cvs version 1.05.

Commit causes Emacs to hang
: Emacs waits for the '`cvs commit`' command to finish before you can do anything. If you start a background job from the loginfo file you must take care that it closes '`stdout`' and '`stderr`' if you do not want to wait for it. (You do that with '`background-command &>- 2&>- &`' if you are starting '`background-command`' from a '`/bin/sh`' shell script).

  Your emacs will also hang if there was a lock file in the repository. In this case you can type `C-g` to get control over your emacs again.

Name clash in Emacs 19
: This is really a bug in Elib or the Emacs 19 distribution. Both Elib and Emacs 19.6 through at least 19.10 contains a file named '`cookie.el`'. One of the files will have to be renamed, and we are currently negotiating about which of the files to rename.

Commands while cvs-update is running
: It is possible to type commands in the *cvs* buffer while the update is running, but error messages is all that you will get. The error messages should be better.

Unexpected output from CVS
: Unexpected output from CVS confuses pcl-cvs. It will currently create a bug report that you can mail to me. It should do something more civilized.

Function and Variable Index

Concept Index

Key Index

Short Contents

Table of Contents