Ediff User's Manual

Ediff version 2.34

July 1995

Michael Kifer

1 Introduction

Ediff provides a convenient way for merging and comparing pairs (or triples) of files and buffers. The files being compared, file-A, file-B, and file-C (if applicable) are shown in separate windows (side by side, one above the another, or in separate frames), and the differences are highlighted as you step through them. You can also copy difference regions from one buffer to another (and recover old differences if you change your mind).

Another powerful feature is the ability to merge a pair of files into a third buffer. Merging with an ancestor file is also supported. Furthermore, Ediff is equipped with directory-level capabilities that allow the user to conveniently launch browsing or merging sessions on groups of files in two (or three) different directories.

In addition, Ediff can apply a patch to a file and then let you step though both files, the patched and the original one, simultaneously, difference-by-difference. You can even apply a patch right out of a mail buffer, i.e., patches received by mail don't even have to be saved. Since Ediff lets you copy differences between buffers, you can, in effect, apply patches selectively (i.e., you can copy a difference region from `file_orig` to `file`, thereby undoing any particular patch that you don't like).

Unfortunately, Ediff still doesn't understand multi-file patches—this requires further work (volunteers needed!).

Ediff is aware of version control, which lets the user compare files with their older versions. Ediff also works with remote and compressed files, automatically ftp-ing them over and uncompressing them. See Chapter 6 [Remote and Compressed Files], page 9, for details.

This package builds upon ideas borrowed from `emerge.el` and several Ediff's routines are adaptations from that package. Although Ediff subsumes `emerge.el` in its functionality, much of that functionality of Ediff is influenced by `emerge.el`. The implementation and the interface are, of course, drastically different.

2 Major Entry Points

Ediff can be invoked interactively using the following functions, which can be run either from the minibuffer or from the menu bar. In the menu bar, all Ediff's entry points belong to three submenus of the 'Tools' menu: 'Compare,' 'Merge,' and 'Apply Patch.' You don't have to remember these entry points, if Ediff is invoked via the menu bar.

`ediff-files`
`ediff` Compare two files.

`ediff-buffers`
Compare two buffers.

`ediff-files3`
`ediff3` Compare three files.

`ediff-buffers3`
Compare three buffers.

`ediff-directories`
`edirs` Compare files common to two directories.

`ediff-directories3`
`edirs3` Compare files common to three directories.

`ediff-directory-revisions`
`edir-revisions`
Compare versions of files in a given directory. Ediff selects only the files that are under version control.

`ediff-merge-directory-revisions`
`edir-merge-revisions`
Merge versions of files in a given directory. Ediff selects only the files that are under version control.

`ediff-merge-directory-revisions-with-ancestor`
`edir-merge-revisions-with-ancestor`
Merge versions of files in a given directory using other versions as ancestors. Ediff selects only the files that are under version control.

`ediff-windows-wordwise`
Compare windows word-by-word.

`ediff-windows-linewise`
Compare windows line-by-line.

`ediff-regions-wordwise`
Compare regions word-by-word.

`ediff-regions-linewise`
Compare regions line-by-line.

`ediff-revision`
Compare versions of current buffer, if the buffer is visiting a file under version control.

`ediff-patch-file`
`epatch` Patch file then compare. At present, doesn't understand multi-file patches.

`ediff-patch-buffer`
`epatch-buffer`
Patch buffer then compare.

`ediff-merge-files`
`ediff-merge`
Merge two files.

`ediff-merge-files-with-ancestor`
`ediff-merge-with-ancestor`
Same but with ancestor.

`ediff-merge-buffers`
Merge two buffers.

`ediff-merge-buffers-with-ancestor`
Same but with ancestor.

`ediff-merge-directories`
`edirs-merge`
Merge files common to two directories.

`ediff-merge-directories-with-ancestor`
`edirs-merge-with-ancestor`
Same but using files in a third directory as ancestors.

`ediff-merge-revisions`
Merge two versions of the file visited by the current buffer. Same but with ancestor.

If you want Ediff to be loaded from the very beginning, you should have

```
(require 'ediff)
```

in your .emacs file. Otherwise, Ediff will be loaded automatically when you use one of the above functions, either directly or through the menus.

When the above functions are invoked, they prompt the user about the information they need—typically the files or buffers to compare or patch. Ediff tries to be smart about these prompts. For instance, in comparing/merging files, it will offer the visible buffers as defaults. In prompting for files, if the user enters a directory, the previously input file name will be appended to that directory. In addition, if the variable `ediff-use-last-dir` is not `nil`, Ediff will offer previously entered directories as defaults (which will be maintained separately for each type of file, A, B, or C).

All the above functions use the Unix `diff` utility to find difference regions. They process `diff` output and display it to the user in a convenient form. At present, Ediff understands only the plain output from diff. Options such as '-c' are not supported, nor is the format produced by VMS diff.

The functions `ediff-files`, `ediff-buffers`, `ediff-files3`, `ediff-buffers3` first display the coarse, line-based difference regions, as reported by the `diff` program. Since `diff` may report fairly large chunks of text as being different, even though the difference may be

localized to just a few words or even to the white space or line breaks, Ediff will further *refine* the regions to indicate which exact words differ. If the only difference is in the white space and line breaks, Ediff will say so. On a color display, fine differences are highlighted with color; on a monochrome display, they are underlined. See Section 7.5 [Highlighting Difference Regions], page 15, to learn how to change that.

The functions `ediff-windows-wordwise`, `ediff-windows-line-wise`, `ediff-regions-wordwise` and `ediff-regions-linewise` do comparison on parts of buffers (which must already exist). Since `ediff-windows-wordwise` and `ediff-regions-wordwise` are intended for relatively small segments of buffers, comparison is done on the basis of words rather than lines. No refinement is necessary in this case. This technique is effective only for relatively small regions (perhaps, up to 100 lines), as these functions have a relatively slow startup.

To compare large regions, use ediff-regions-linewise. In this mode, Ediff displays differences as it would if invoked via `ediff-files` or `ediff-buffers.`

The functions `ediff-patch-file` and `ediff-patch-buffer` apply a patch to a file or a buffer and then run Ediff on these buffers, displaying the difference regions. Currently, Ediff still doesn't understand multi-file patches (volunteers?).

The entry points `ediff-directories`, `ediff-merge-directories`, etc., provide a convenient interface for comparing and merging files in different directories. The user is presented with Dired-like interface from which one can run a group of related Ediff sessions.

Finally, for files under version control, `ediff-revision` lets the user compare versions of the file visited by the current buffer. Moreover, the functions `ediff-directory-revisions`, `ediff-merge-directory-revisions`, etc., let the user run a group of related Ediff sessions by taking a directory and comparing (or merging) versions of files in that directory (for files that are under version control).

3 Commands

All Ediff commands pertinent to a given session are displayed in a quick help window, unless you type `?` to shrink the window to just one line. You can redisplay the help window by hitting `?` again. In this section we comment only on the features that cannot be readily deduced from the quick help window. You can always type `E` in any control window to bring up this manual.

Many Ediff commands take numeric prefix arguments. For instance, if you type a number, say 3, and then `j` (ediff-jump-to-difference), Ediff will take you to the 3d difference region. Typing 3 and then `a` (`ediff-diff-to-diff`) will copy the 3d difference region from buffer A to buffer B. Hitting `b` does copying in the opposite direction. (In 3-way comparison mode, the commands for copying are `ab`, `ba`, `ca`, etc.) Likewise, 4 followed by `ra` will restore the 4th difference region in buffer A (if it was previously saved as a result of copying from, say, buffer B to A).

Without the prefix argument, all commands operate on the current difference region.

The total number of differences and the current difference number are always displayed in the mode line of the control window.

If, after making changes to buffers A, B, or C, you decide to save them, it is recommended to use `ediff-save-buffer`, which is bound to `wa`, `wb`, and `wc` (`wa` will save buffer A, `wb` saves buffer B, etc.).

Typing `wd` saves the output from the `diff` utility to a file, so you can later refer to it. With prefix argument, this command saves the plain output from `diff` (see `ediff-diff-program` and `ediff-diff-options`). Without the argument, it saves customized `diff` output (see `ediff-custom-diff-program` and `ediff-custom-diff-options`), if it is available.

Instead of saving it, `diff` output can be *displayed* using the command `D`. Without the prefix argument, it displays the customized `diff` output of the session. With the prefix argument, it displays the plain `diff` output If either of the `diff` outputs is unavailable (because it wasn't generated or the user killed the respective buffer), then Ediff will try to display the other `diff` output. If none is available, a warning is issued.

The command `z` suspends the current ediff session. It hides the control buffer and the variants. The easiest way to resume a suspended Ediff session is through the registry of active sessions. See Chapter 4 [Registry of Ediff Sessions], page 7, for details.

The command `q` quits the current Ediff session. With a prefix argument, it will ask the user whether to delete the variant buffers.

The command `s` is used only for merging. It allows the user to shrink window C to its minimal size, thereby exposing as much of buffers A and B as possible. This command is intended only for temporary viewing. Therefore, Ediff will restore the original window size for buffer C whenever window configuration is changed by the user (on toggling the help, changing the manner in which windows are split, etc.). However, recentering and jumping to a difference does not affect window C. Typing `s` again restores the original size of the merge window.

With a positive prefix argument, the command `s` makes the merge window, window C, slightly taller. With `-` or a negative prefix argument, `s` makes window C slightly shorter.

In the merge mode, Ediff uses a default variant (one of the two files being merged) when it cannot decide which of the difference regions (that of buffer A or buffer B) should be copied into the merge buffer. A user may decide that the default variant was chosen inappropriately and may wish to change that while merging is in progress. To facilitate this, Ediff has a command, bound to `&`, which will cause Ediff to start merging anew beginning with the current difference, using an alternative default variant (the user is asked to type in the new default for merging), which can be either 'default-A', 'default-B', or 'combined'. See Section 7.9 [Merging and diff3], page 18, for details.

Such repeated merging affects only difference regions that have default-A/B status, and only if they were not changed with respect to their originals.

Another command that is used only for merging is `+`. Its effect is to combine the current difference regions of buffers A and B and put the combination into the merge buffer. See Section 7.9 [Merging and diff3], page 18, specifically, the variables `ediff-combine-diffs` and `ediff-combination-pattern`.

Some commands are not bound to any key:

`ediff-show-registry`
`eregistry`
: This command brings up the registry of active Ediff sessions. Ediff registry is a useful device that can be used for resuming Ediff sessions when the user switched to some other work before finishing a comparison or merging job. It is also useful for switching between multiple active Ediff sessions that are run at the same time. The function `eregistry` is an alias for `ediff-show-registry`.

`ediff-toggle-multiframe`
: Changes the display from the multi-frame mode (where the quick help window is in a separate frame) to the single-frame mode (where all Ediff buffers share the same frame), and vice versa.

`ediff-revert-buffers-then-recompute-diffs`
: This is useful when, after making changes, you decided to make a fresh start, or if at some point you changed the files being compared but want to discard any changes to comparison buffers that were done since then. This command will ask for confirmation before reverting files. With a prefix argument, it will revert files without asking.

`ediff-profile`
: Ediff has an admittedly primitive (but useful) facility for profiling Ediff's commands. Users should not be concerned with this feature, unless they are willing to put time into improving the efficiency of Ediff. The function `ediff-profile` toggles profiling of ediff commands.

4 Registry of Ediff Sessions

Ediff maintains a registry of all its invocations that are still *active*. This feature is very convenient for switching among active Ediff sessions or for quickly restarting a suspended Ediff session.

The focal point of this activity is a buffer called **Ediff Registry**. You can display this buffer by typing `R` in any Ediff Control Buffer or Session Group Buffer (see Chapter 5 [Session Groups], page 8), or by typing `M-x eregistry` into the Minibuffer. The latter would be the fastest way to bring up the registry buffer if no control or group buffer is displayed in any of the visible Emacs windows. If you are in a habit of running multiple long Ediff sessions and often need to suspend, resume, or switch between them, it may be a good idea to have the registry buffer permanently displayed in a separate, dedicated window.

The registry buffer has several convenient key bindings. For instance, clicking mouse button 2 or typing `RET` or `v` over any session record resumes that session. Session records in the registry buffer provide a fairly complete description of each session, so it is usually easy to identify the right session to resume.

Other useful commands are bound to `SPC` (next registry record) and `DEL` (previous registry record). There are other commands as well, but you don't need to memorize them, since they are listed at the top of the registry buffer.

5 Session Groups

Several major entries of Ediff perform comparison and merging on directories. On entering `ediff-directories`, `ediff-directories3`, `ediff-merge-directories`, `ediff-merge-directories-with-ancestor`, `ediff-directory-revisions`, `ediff-merge-directory-revisions`, or `ediff-merge-directory-revisions-with-ancestor`, the user is presented with a Dired-like buffer that lists files common to the directories involved along with their sizes. (The list of common files can be further filtered through a regular expression, which the user is prompted for.) We call this buffer *Session Group Panel* because all Ediff sessions associated with the listed files will have this buffer as a common focal point.

Clicking button 2 or typing *RET* or *v* over a record describing files invokes Ediff in the appropriate mode on these files. You can come back to the session group buffer associated with a particular invocation of Ediff by typing *M* in Ediff control buffer of that invocation.

Many commands are available in the session group buffer; some are applicable only for certain types of work. The relevant commands are always listed at the top of each session group buffer, so there is no need to memorize them.

In directory comparison or merging, a session group panel displays only the files common to all directories involved. The differences are kept in a separate buffer and are conveniently displayed by typing *D* to the corresponding session group panel. Thus, as an added benefit, Ediff can be used to compare the contents of up to three directories.

Session records in session group panels are also marked with `+`, for active sessions, and with `-`, for finished sessions.

Sometimes, it is convenient to exclude certain session records from a group. Usually this happens when the user doesn't intend to run Ediff of certain files in the group, and the corresponding session records just add clutter to the session group buffer. To help alleviate this problem, the user can type *x* to mark a session as a candidate for exclusion and *x* to actually hide the marked sessions. There actions are reversible: with a prefix argument, *h* unmarks the session under the cursor, and *x* brings the hidden sessions into the view (*x* doesn't unmark them, though, so the user has to explicitly unmark the sessions of interest).

Group sessions also understand the command *m*, which marks sessions for future operations (other than hiding) on a group of sessions. At present, the only such group-level operation is the creation of a multi-file patch.

A multi-file patch is a concatenated output of several runs of the Unix `diff` command (some versions of `diff` let you create a multi-file patch in just one run). In a session group buffer created in response to `ediff-directories` or `ediff-directory-revisions`, the user can type *P* to create a multi-file patch of marked sessions (which must be marked using the *m* command). Ediff then will display a buffer containing the patch. In an `ediff-directories` session, it is enough to just mark the requisite sessions. In `ediff-directory-revisions` revisions, the marked sessions must also be active, or else Ediff will refuse to produce a multi-file patch. This is because, in the latter-style sessions, there are many ways to create diff output, and it is easier to handle by running Ediff on the inactive sessions.

6 Remote and Compressed Files

Ediff works with remote, compressed, and encrypted files. Ediff supports ange-ftp.el, jka-compr.el, uncompress.el and crypt++.el, but it may work with other similar packages as well. This means that you can compare files residing on another machine, or you can apply a patch to a file on another machine (even the patch itself can be a remote file!).

When patching compressed or remote files, Ediff does not rename the source file (unlike what the `patch` utility would usually do). Instead, the source file retains its name and the result of applying the patch is placed in a temporary file that has the suffix `_patched` attached. Generally, this applies to files that are handled using black magic, such as special file handlers (ange-ftp and some compression and encryption packages all use this method).

Regular files are treated by the `patch` utility in the usual manner, i.e., the original is renamed into `source-name_orig` and the result of the patch is placed into the file source-name. (Ediff uses `_orig` instead of the usual `.orig` to placate systems like VMS.)

7 Customization

Ediff has a rather self-explanatory interface, and in most cases the user won't need to change anything. However, should the need arise, there are extensive facilities to change the default behavior.

Most of the customization can be done by setting various variables in the `.emacs` file. Some customization (mostly window-related customization and faces) can be done by putting appropriate lines in `.Xdefaults`, `.xrdb`, or whatever X resource file is in use.

With respect to the latter, it is important to be aware that the X resource for Ediff customization is 'Ediff', *not* 'emacs'. See Section 7.3 [Window and Frame Configuration], page 12, See Section 7.5 [Highlighting Difference Regions], page 15, for further details. Please also refer to Emacs manual for the information on how to set Emacs X resources.

7.1 Hooks

The bulk of customization can be done via the following hooks:

`ediff-load-hooks`
: Can be used to change defaults after Ediff is loaded. These hooks are executed right after the default bindings are set.

`ediff-keymap-setup-hooks`
: Can be used to alter bindings in Ediff's keymap. These hooks are called right after the default bindings are set.

`ediff-before-setup-windows-hooks`
`ediff-after-setup-windows-hooks`
: Called before/after Ediff sets up its window configuration. Can be used to save the configuration that existed before Ediff starts or for whatever other purposes.

`ediff-suspend-hooks`
`ediff-quit-hooks`
: Can be used to set desired window configurations, delete files Ediff didn't want to clean up after exiting, etc. By default, `ediff-quit-hooks` is set to a function, `ediff-cleanup-mess`, which cleans after Ediff, as appropriate in most cases. It is rather unlikely that the user will want to change it. However, the user may want add other hooks to `ediff-quit-hooks`, either before or after `ediff-cleanup-mess` (see the documentation for `add-hook` in Emacs manual on how to do this). One should be aware that hooks executing before `ediff-cleanup-mess` start in `ediff-control-buffer;` they should also leave `ediff-control-buffer` as the current buffer when they finish. Hooks that are executed after `ediff-cleanup-mess` should expect the current buffer be either buffer A or buffer B. `ediff-cleanup-mess` doesn't kill the buffers being compared or merged (see `ediff-cleanup-hooks`, below).

`ediff-cleanup-hooks`
: Default is `nil`. Hooks to run just before running `ediff-quit-hooks`. This is a good place to do various cleanups, such as deleting the variant buffers. Ediff provides a function, `ediff-janitor`, as one such possible hook, which the user

can `add-hooks` to `ediff-cleanup-hooks`. This function kills buffers A, B, and, possibly, C, if these buffers aren't modified. In merge jobs, buffer C is never deleted. However, the side effect of using this function is that you may not be able to compare the same buffer in two separate Ediff sessions: quitting one of them will delete this buffer in another session as well.

`ediff-before-setup-control-frame-hooks`
`ediff-after-setup-control-frame-hooks`
: Can be used to relocate Ediff control frame when Ediff runs in a multiframe mode (i.e., when the control buffer is in its own dedicated frame). Be aware that many variables that drive Ediff are local to Ediff Control Panel (`ediff-control-buffer`), which requires special care in writing these hooks. Take a look at `ediff-default-suspend-hook` and `ediff-default-quit-hook` to see what's involved.

`ediff-startup-hooks`
: Last hook called after Ediff starts up.

`ediff-select-hooks`
: Called after Ediff selects the next difference region.

`ediff-unselect-hooks`
: Called after Ediff unselects the current difference region.

`ediff-prepare-buffer-hooks`
: Hooks executed for each Ediff buffer (A, B, C) right after these buffers are arranged.

`ediff-display-help-hooks`
: Ediff executes these hooks each time after setting up the help message. Can be used to alter the help message for custom packages that run on top of Ediff.

`ediff-mode-hooks`
: Called just after Ediff mode is set up in the control buffer. This is done before any Ediff window is created. One can use it to set local variables that alter the look of the display.

`ediff-registry-setup-hooks`
: Hooks run after setting up the registry for all active Ediff session. See Chapter 5 [Session Groups], page 8, for details.

`ediff-session-group-setup-hooks`
: Hooks run after setting up a control panel for a group of related Ediff sessions. See Chapter 5 [Session Groups], page 8, for details.

7.2 Quick Help

Ediff provides quick help using its control panel window. Since this window takes a fair share of the screen real estate, you can toggle it off by hitting `?`. The control window will then shrink to just one line and a mode line, displaying a short help message. The variable `ediff-prefer-long-help-message` tells Ediff whether the user wants the short message initially or the long one. By default, it is set to `nil`, meaning that the short message will

be shown on startup. Set this to `t`, if you want Ediff to start with the long message. If you want to change the appearance of the help message on a per-buffer basis, you must use `ediff-startup-hooks` to change the value of the variable `ediff-help-message`, which is local to `ediff-control-buffer`.

7.3 Window and Frame Configuration

On a non-windowing display, Ediff sets things up in one frame, splitting it between a small control window and the windows for buffers A, B, and C. The split between these windows can be horizontal or vertical, which can be changed interactively by typing `|` while the cursor is in the control window.

On a window display, Ediff sets up a dedicated frame for Ediff Control Panel and then it chooses windows as follows: If one of the buffers is invisible, it is displayed in the currently selected frame. If a buffer is visible, it is displayed in the frame where it is visible. If, according to the above criteria, the two buffers fall into the same frame, then so be it—the frame will be shared by the two. The same algorithm works when you hit *C-l* (`ediff-recenter`), *p* (`ediff-previous-difference`), *n* (`ediff-next-difference`), etc.

The above behavior also depends on whether the current frame is splittable, dedicated, etc. Unfortunately, the margin is too small to present this remarkable algorithm.

The bottom line of all this is that you can compare buffers in one frame or in different frames. The former is done by default, while the latter can be achieved by arranging buffers A, B (and C, if applicable) to be seen in different frames. Ediff respects these arrangements, automatically adapting itself to the multi-frame mode.

Ediff uses the following variables to set up its control panel (a.k.a. control buffer, a.k.a. quick help window):

`ediff-control-frame-parameters`
: The user can change or augment this variable including the font, color, etc. The X resource name of Ediff Control Panel frames is 'Ediff'. Under X-windows, you can use this name to set up preferences in your `~/.Xdefaults`, `~/.xrdb`, or whatever X resource file is in use. Usually this is preferable to changing `ediff-control-frame-parameters` directly. For instance, you can specify in `~/.Xdefaults` where the control frame is to be sitting on the screen using the resource `Ediff*geometry`.

  In general, any X resource pertaining the control frame can be reached via the prefix `Ediff*`.

`ediff-control-frame-position-function`
: The prefered way of specifying the position of the control frame is by setting the variable `ediff-control-frame-position-function` to an appropriate function. The default value of this variable is `ediff-make-frame-position`. This function places the control frame in the vicinity of the North-East corner of the frame displaying buffer A.

The following variables can be used to adjust the location produced by `ediff-make-frame-position` and for related customization.

`ediff-narrow-control-frame-leftward-shift`
: Specifies the number of characters for shifting the control frame from the rightmost edge of frame A when the control frame is displayed as a small window.

`ediff-wide-control-frame-rightward-shift`
: Specifies the rightward shift of the control frame from the left edge of frame A when the control frame shows the full menu of options.

`ediff-control-frame-upward-shift`
: Specifies the number of pixels for the upward shift of the control frame.

`ediff-prefer-iconified-control-frame`
: If t, the control frame becomes iconified automatically when the quick help message is toggled off. This saves valuable real estate on the screen. Toggling help back will deiconify the control frame.

  To start Ediff with an iconified Control Panel, you should set this variable to t and `ediff-prefer-long-help-message` to nil. This behavior is useful only in Emacs (not in XEmacs) and only if the window manager is TWM or a derivative.

If you truly and absolutely dislike the way Ediff sets up windows and if you cannot change this via frame parameters, the last resort is to rewrite the function `ediff-setup-windows`. However, we believe that detaching Ediff Control Panel from the rest and making it into a separate frame offers an important opportunity by allowing you to iconify that frame. Under Emacs, the icon will usually accept all of the Ediff commands, but will free up valuable real estate on your screen (this may depend on the window manager, though). Iconifying won't do any good under XEmacs since XEmacs icons are not sensitive to keyboard input. The saving grace is that, even if not iconified, the control frame is very small, smaller than some icons, so it does not take much space in any case.

The following variable controls how windows are set up.

`ediff-window-setup-function`
: The multiframe setup is achieved via `ediff-setup-windows-multiframe` function, which is a default on windowing displays. The plain setup, one where all windows are always in one frame, is done via `ediff-setup-windows-plain`, which is the default on a non-windowing display (or in an xterm window). In fact, under Emacs, you can switch freely between these two setups by executing the command `ediff-toggle-multiframe` using the Minibuffer.

  If you don't like any of these setups, write your own function. See the documentation for `ediff-window-setup-function` for the basic guidelines. However, writing window setups is not easy, so before embarking on this job you may want to take a close look at `ediff-setup-windows-plain` and `ediff-setup-windows-multiframe`.

The user can run multiple Ediff sessions at once, by invoking Ediff several times without exiting previous Ediff sessions. Different sessions may even operate on the same pair of files. Each session would have its own Ediff Control Panel and all the regarding a particular session is local to the associated control panel buffer. You can switch between sessions by suspending one session and then switching to another control panel. (Different control panel buffers are distinguished by a numerical suffix, e.g., Ediff Control Panel<3>.)

7.4 Selective Browsing

Sometimes it is convenient to be able to step through only some difference regions, those that satisfy certain conditions, and to ignore all others. The commands *#f* and *#h* let the user specify regular expressions to control the way Ediff skips to the next or previous difference. Typing *#f* lets one specify regular expressions for each buffer, regexp-A, regexp-B, and regexp-C. Ediff will then start stepping through only those difference regions where the region in buffer A matches regexp-A and/or the region in buffer B matches regexp-B, etc. Whether 'and' or 'or' will be used depends on how the user responds to a prompt. Similarly, using *#h*, one specifies expressions that match difference regions to be ignored while stepping through the differences. That is, if the buffer A part matches regexp-A, the buffer B part matches regexp B and (if applicable) buffer-C part matches regexp-C, then the region will be ignored by the commands *n*/*SPC* (`ediff-next-difference`) and *p*/*DEL* (`ediff-previous-difference`) commands.

Hitting *#f* and *#h* toggles selective browsing on/off.

Note that selective browsing affects only `ediff-next-difference` and `ediff-previous-difference`, i.e., the commands invoked by typing *n*/*SPC* and *p*/*DEL*. You can still jump directly (using *j* or *ga*/*gb*/*gc*) to any numbered difference. Also, it should be understood, that *#f* and *#h* do not change the position of the point in the buffers. The effect of these commands is seen only when the user types *n* or *p*, i.e., when Ediff is told to jump to the next or previous difference.

Users can supply their own functions to specify how Ediff should do selective browsing. To change the default Ediff function, add a function to `ediff-load-hooks` which will do the following assignments:

```
(fset ediff-hide-regexp-matches 'your-hide-function)
(fset ediff-focus-on-regexp-matches 'your-focus-function)
```

Useful hint: To specify a regexp that matches everything, don't simply type *RET* in response to a prompt. Typing *RET* tells Ediff to accept the default value, which may not be what you want. Instead, you should enter something like '`^`' or '*$*' — which matches every line.

If the user does not remember if selective browsing is in effect and which regexps are being used, the status command, *i*, will supply the requisite information.

In addition to the ability to ignore regions that match regular expressions, Ediff can be ordered to start skipping over certain 'inessential' regions. This is controlled by the variable

`ediff-ignore-similar-regions`
: If `t`, causes Ediff to skip over difference regions that deemed inessential, i.e., where the only differences are those in the white space and newlines.

Note: In order for this feature to work, auto-refining of difference regions must be on, since otherwise Ediff won't know if there are fine differences between regions. Under X, auto-refining is a default, but it is nixed on dumb terminals or in Xterm windows. Therefore, in a non-windowing environment, the user must explicitly turn auto-refining on (e.g., by typing `@`).

Caution: If many inessential regions appear in a row, Ediff may take a long time to jump to the next region because it has to compute fine differences of all intermediate regions.

7.5 Highlighting Difference Regions

The following variables control the way Ediff highlights difference regions.

`ediff-before-flag-bol`
`ediff-after-flag-eol`
`ediff-before-flag-mol`
`ediff-after-flag-mol`

The above are ASCII strings that mark the beginning and the end of the differences found in files A, B, and C. Ediff uses different flags to highlight regions that begin/end at the beginning/end of a line or in a middle of a line.

`ediff-current-diff-face-A`
`ediff-current-diff-face-B`
`ediff-current-diff-face-C`

Ediff uses these faces to highlight current differences on X displays. These and subsequently described faces can be set either in `.emacs` or in `.Xdefaults`. The X resource for Ediff is 'Ediff', *not* 'emacs'. Please refer to Emacs manual for the information on how to set X resources.

`ediff-fine-diff-face-A`
`ediff-fine-diff-face-B`
`ediff-fine-diff-face-C`

Faces used to show the fine differences between the current differences regions in buffers A, B, and C, respectively.

`ediff-even-diff-face-A`
`ediff-even-diff-face-B`
`ediff-even-diff-face-C`
`ediff-odd-diff-face-A`
`ediff-odd-diff-face-B`
`ediff-odd-diff-face-C`

Non-current difference regions are displayed using these alternating faces. The odd and the even faces are actually identical on monochrome displays, because without colors options are limited. So, Ediff uses italics to highlight non-current differences.

`ediff-highlight-all-diffs`

Indicates whether—on a window system—the user wants differences to be marked using ASCII strings (like on a dumb terminal) or using colors and highlighting. Normally, Ediff highlights all differences, but the selected difference is highlighted more visibly. One can cycle through various modes of highlighting by hitting *h*. By default, Ediff starts in the mode where all difference regions are highlighted. If you prefer to start in the mode where unselected differences are not highlighted, you should set `ediff-highlight-all-diffs` to `nil`. Typing *h* restores highlighting of all differences.

Ediff lets you switch between the two modes of highlighting. That is, you can switch interactively from highlighting using faces to highlighting using ASCII flags, and back. Of course, switching has effect only under a windowing sys-

tem. On a dumb terminal or in an xterm window, the only available option is highlighting with ASCII flags.

If you want to change the above variables, they must be set **before** Ediff is loaded.

There are two ways to change the default setting for highlighting faces: either change the variables, as in

```
(setq ediff-current-diff-face-A 'bold-italic)
```

or

```
(setq ediff-current-diff-face-A
(copy-face 'bold-italic 'ediff-current-diff-face-A))
```

or modify the defaults selectively:

```
(add-hook 'ediff-load-hooks
          (function (lambda ()
                      (set-face-foreground ediff-current-diff-face-B "blue")
                      (set-face-background ediff-current-diff-face-B "red")
                      (make-face-italic ediff-current-diff-face-B))))
```

You may also want to take a look at how the above faces are defined in the source code of Ediff.

Note: it is not recommended to use `internal-get-face` (or `get-face` in XEmacs) when defining Ediff's faces, since this may cause problems when there are several frames with different font sizes. Instead, use `copy-face`, `set-face-*`, or `make-face-*` as shown above.

7.6 Narrowing

If buffers being compared are narrowed at the time of invocation of Ediff, `ediff-buffers` will preserve the narrowing range. However, if `ediff-files` is invoked on the files visited by these buffers, narrowing will be turned off, since we assume that the user wants to compare the entire files.

Invocation of `ediff-regions-wordwise/linewise` and `ediff-windows-wordwise/linewise` will cause Ediff to set new narrowing ranges (corresponding to the windows being compared). However, the old ranges are preserved and can be restored by typing `%`. The original ranges will be also restored on quitting Ediff.

Two variables control the behavior of `ediff-windows-wordwise/linewise`, `ediff-regions-wordwise/linewise` with respect to narrowing:

`ediff-start-narrowed`
: If `t`, Ediff will narrow the display to the appropriate range if it is invoked as `ediff-windows-wordwise/linewise` or `ediff-regions-wordwise/linewise.` If `nil`, narrowing will not take place. However, the user can still toggle narrowing on and off by typing `%`.

`ediff-quit-widened`
: Controls whether on exiting Ediff should restore the visibility range that existed before the current invocation.

7.7 Refinement of Difference Regions

Ediff has variables to control the way fine differences are highlighted. This feature give the user control over the process of refinement. Note that refinement ignores spaces, tabs, and newlines.

`ediff-auto-refine`
: The default is 'on', which means that fine differences within regions will be highlighted automatically. On a slow machine, automatic refinement may be painful. In that case, the user can toggle auto-refining on or off (or nix it completely) by hitting `@`. When auto-refining is off, fine differences will be shown only for regions for which these differences have been computed and saved before. If auto-refining is nixed, fine differences will not be shown at all. Hitting `*` will compute and redisplay fine differences for the current difference region, regardless of the status auto-refining.

`ediff-auto-refine-limit`
: If auto-refining is on, this variable limits the size of the regions to be auto-refined. This guards against the possible slow-down that may be caused by extraordinary large difference regions. The user can always refine the current region by typing `*`.

`ediff-forward-word-function`
: Gives the user control over how fine differences are computed. The value must be a lisp function that determines how the current difference region should be split into words.

  Fine diferences are computed by first splitting the current difference region into words and then passing this along to `ediff-diff-program`. For the default `ediff-forward-word-function` (which is `ediff-forward-word`), a word is a string consisting of letters, '-', or '_'; a string of punctuation symbols; a string of digits, or a string consisting of symbols that are neither space, nor a letter.

  This default behavior is controlled by four variables: `ediff-word-1`, ..., `ediff-word-4`. See the on-line documentation for these variables and for the function `ediff-forward-word` for an explanation of how to modify these variables.

Sometimes, when a region has too many differences between the variants, highlighting of fine differences stands in the way, especially on color displays. If that is the case, the user can type `*` with a negative prefix argument, which would unhighlight fine differences for the current region.

To unhighlight fine differences in all `diff` regions, use the command `@`. Repeated typing of this key cycles through three different states: auto-refining, no-auto-refining, and no-highlighting of fine differences.

7.8 Patch and Diff Programs

The next group of variables determines the programs to be used for applying patches and for computing the main difference regions (not the fine difference regions):

`ediff-patch-program`
`ediff-diff-program`
`ediff-diff3-program`
: Specify the functions that produce differences and do patching.

`ediff-patch-options`
`ediff-diff-options`
`ediff-diff3-options`
: Specify which options to pass to the above utilities. It is unlikely that you would want to change these. However, sometimes you may want to tell `diff` to ignore spaces and such. Use the option '-w' for that. Diff has several other useful options (type 'man diff' to find out). However, Ediff does not let you use the option '-c', as it doesn't recognize this format yet. If you need to save the output from `diff` in a special format, Ediff lets you specify "custom" `diff` format using the following two variables:

`ediff-custom-diff-program`
: The output generated by `ediff-custom-diff-program` (which doesn't even have to be a Unix-style `diff`!) is not used by Ediff. It is provided exclusively so that the user could save if using the function `ediff-save-buffer` (normally bound to *`wd`*) and later refer to it. However, Ediff is not the preferred way of producing `diff` output in Emacs. Unless you also intend to use Ediff for browsing through the diff'ed files, *`M-x diff`* may be a faster way to generate output from `diff`.

`ediff-custom-diff-options`
: Specifies the options to pass to `ediff-custom-diff-program`.

Beware of VMS Diff: The output from VMS Diff is not yet supported. Instead, make sure some implementation of Unix diff, such as gnudiff, is used.

7.9 Merging and diff3

Ediff supports 3-way comparison via the functions `ediff-files3` and `ediff-buffers3`. The interface is the same as for 2-way comparison. In 3-way comparison and merging, Ediff reports if any two difference regions are identical. For instance, if the current region in buffer A is the same as the region in buffer C, then the mode line of buffer A will display [=diff(C)] and the mode line of buffer C will display [=diff(A)].

Merging is done according to the following algorithm.

If a difference region in one of the buffers, say B, differs from the ancestor file while the region in the other buffer, A, doesn't, then the merge buffer, C, gets B's region. Similarly when buffer A's region differs from the ancestor and B's doesn't.

If both regions in buffers A and B differ from the ancestor file, Ediff will choose the region according to the value of the variable

`ediff-default-variant`
: If set to 'default-A' then A's region is chosen. If set to 'default-B' then B's region is chosen. If set to 'combined' then the region in buffer C will look like this:

```
#ifdef NEW  /* variant A */
```

```
difference region from buffer A
#else  /* variant B */
difference region from buffer B
#endif  /* NEW */
```

The actual strings that separate the regions copied from bufer A and B are controlled by the variable

`ediff-combination-pattern`
: A list of three strings. The first is inserted before the difference region of buffer A; the second string goes between the regions; the third will trail region B, as shown in the above example.

In addition to the state of the difference, during merging Ediff displays the state of the merge for each region. If a difference came from buffer A by default (because both regions A and B were different from the ancestor and ediff-default-variant was set to 'default-A') then [=diff(A) default-A] is displayed in the mode line. If the difference in buffer C came, say, from buffer B because the difference region in that buffer differs from the ancestor, but the region in buffer A does not (if merging with an ancestor) then [=diff(B) prefer-B] is displayed. The indicators default-A/B and prefer-A/B are inspired by emerge.el and have the same meaning.

Another indicator of the state of merge is 'combined'. It appears with any difference region in buffer C that was obtained by combining the difference regions in buffers A and B as explained above.

In addition to state of merge and difference indicator, in merging with an ancestor file or buffer, Ediff informs the user when the current difference region in the (normally invisible) ancestor buffer is empty via the *AncestorEmpty* indicator. This helps determine if the changes made to the original in variants A and B represent pure insertion or deletion of text: if the mode line shows *AncestorEmpty* and the corresponding region in buffers A or B is not empty, this means that new text was inserted. If this indicator is not present and the difference regions in buffers A or B are non-empty, this means that text was modified. Otherwise, the original text was deleted.

Although the ancestor buffer is normally invisible, Ediff maintains difference regions there and advances the current difference region accordingly. All highlighting of difference regions is provided in the ancestor buffer, except for the fine differences. Therefore, if desired, the user can put the ancestor buffer in a separate frame and watch it there. However, on a TTY, only one frame can be visible at any given time, and Ediff doesn't support any single-frame window configuration where all buffers, including the ancestor buffer, would be visible. However, the ancestor buffer can be displayed by typing `/` to the control window. (Type `C-l` to hide it again.)

Note that the state-of-difference indicators '=diff(A)' and '=diff(B)' above are not redundant, even in the presence of a state-of-merge indicator. In fact, the two serve different purposes. For instance, if the mode line displays [=diff(B) prefer(B)] and you copy a difference region from buffer A to buffer C then '=diff(B)' will change to 'diff-A' and the mode line will display [=diff(A) prefer-B]. This indicates that the difference region in buffer C is identical to that in buffer A, but originally buffer C's region came from buffer B. This is useful to know because the original difference region in buffer C can be recovered by typing `r`, if necessary.

Ediff never changes the state-of-merge indicator, except in response to the `!` command (see below), in which case the indicator is lost. On the other hand, the state-of-difference indicator is changed automatically by the copying/recovery commands, `a`, `b`, `r`, `+`.

If Ediff is asked to recompute differences via the command `!`, the information about origins of the regions in the merge buffer (default-A, prefer-B, or combined) will be lost. This is because recomputing differences in this case means running diff3 on buffers A, B, and the merge buffer, not on the ancestor buffer. (It makes no sense to recompute differences using the ancestor file, since in the merging mode Ediff assumes that the user did not edit buffers A and B, but he may have edited buffer C, and these changes are to be preserved.) Since some difference regions may disappear as a result of editing buffer C and others may arise, there is generally no simple way to tell where the various regions in the merge buffer came from. In fact, recomputing differences erases all information about the ancestor buffer, so it will be unhighlighted and disconnected from the current Ediff session. (However, this doesn't kill the ancestor buffer.)

In 3-way comparison, Ediff tries to disregard regions that consist entirely of white space. For instance, if, say, the current region in buffer A consists of the white space only (or if it is empty), Ediff will not take it into account for the purpose of computing fine differences. The result is that Ediff can provide a better visual information regarding the actual fine differences in the non-white regions in buffers B and C. Moreover, if the regions in buffers B and C differ in the white space only, then a message to this effect will be displayed.

In the merge mode, the share of the split between window C (the window displaying the merge-buffer) and the windows displaying buffers A and B is controlled by the variable

`ediff-merge-window-share`

> The default is 0.5. To make the merge-buffer window smaller, reduce this amount. It is not recommended to increase the size of the merge-window to more than half the frame (i.e., to increase the value of `ediff-merge-window-share`) to more than 0.5, since it would be hard to see the contents of buffers A and B.

The user can temporarily shrink the merge window to just one line by typing `s`. This change is temporary, until Ediff finds a reason to redraw the screen. Typing `s` again restores the original window size.

With a positive prefix argument, this command will make the merge window slightly taller. This change is persistent. With '`-`' or with a negative prefix argument, the command `s` makes the merge window slightly shorter. This change also persistent.

Ediff lets the user automatically ignore the regions where one of the buffer's regions is prefered because it disagrees with the ancestor, while the other buffer agrees with the ancestor. In this case, Ediff displays only the difference regions where the two buffers, A and B, both differ from the ancestor file. The variable that controls this behavior is

`ediff-show-clashes-only`

> The value of this variable can be toggled interactively, by typing *$*. Note that this variable controls only how Ediff chooses the next/previous difference to show. The user can still jump directly to any difference using the command `j` (with a prefix argument specifying the difference number).

7.10 Support for Version Control

Ediff supports version control via the packages `vc.el` and `rcs.el`. The latter is a package written by Sebastian Kremer <sk@thp.Uni-Koeln.DE>, which is available in

```
ftp.cs.buffalo.edu:pub/Emacs/rcs.tar.Z
ftp.uni-koeln.de:/pub/gnu/emacs/rcs.tar.Z
```

To specify which version control package you are using, set the variable `ediff-version-control-package`, e.g.,

```
(setq ediff-version-control-package 'rcs)
```

A symbol. The default, is 'vc'. **Note**: both packages provide access to RCS, but only `vc.el` comes standard with Emacs and XEmacs.

`ediff-revision-key`

A string. For files under revision control, one key can be bound to the function `ediff-revision`, which runs Ediff comparing versions of the current buffer. This is controlled by the above variable. The default is `nil`, i.e., Ediff doesn't bind any key to run `ediff-revision`.

If the version control package used is `vc.el`, then `ediff-revision-key` is bound in a key map accessible through the prefix *C-x v*, i.e., if you have e.g., `(setq ediff-revision-key "=")` in your `~/.emacs` file, then to run `ediff-revision` you will have to type *C-x v =*.

If the version control package is `rcs.el` is used, then the key is bound in the global Emacs map, the one available by default. For that reason, it is recommended that the key should start with a prefix, such as *C-c*. For instance, if you would like to use *C-c E* to run `ediff-revision`, put `(setq ediff-revision-key "\C-cE")` in your `~/.emacs` file.

Note: Ediff doesn't bind `ediff-revision-key` when it is first loaded. The binding takes effect only when the user invokes `ediff-revision`. If you want the binding to take effect right from the start, put this in your `~/emacs`:

```
(setq ediff-revision-key "your-key")
(require 'ediff)
(ediff-load-version-control)
```

If you want the binding to take effect only after Ediff is first loaded into your Emacs, use `ediff-load-hooks`:

```
(setq ediff-revision-key "your-key")
(add-hook 'ediff-load-hooks 'ediff-load-version-control)
```

7.11 Customizing the Mode Line

When Ediff is running, the mode line of Ediff Control Panel buffer shows the current difference number and the total number of difference regions in the two files.

The mode line of the buffers being compared displays the type of the buffer ('A:', 'B:', or 'C:') and (usually) the file name. Ediff is trying to be intelligent in choosing the mode line buffer identification. In particular, it works well with uniquify.el and mode-line.el packages (which improve on the default way in which Emacs displays buffer identification). If you don't like the way Ediff changes the mode line, there always is `ediff-prepare-buffer-hooks`, which can be used to modify the mode line.

7.12 Miscellaneous

The following is the last batch of variables that can be customized:

`ediff-split-window-function`
: Controls the way you want the window be split between file-A and file-B (and file-C, if applicable). It defaults to the vertical split (`split-window-vertically`, but you can set it to `split-window-horizontally`, if you so wish.

`ediff-merge-split-window-function`
: Controls how windows are split between buffers A and B in the merge mode.

`ediff-make-wide-display-function`
: The user can toggle wide/regular display by typing *m*. In the wide display mode, buffers A, B (and C, when applicable) are displayed in a single frame that is as wide as the entire workstation screen. This is useful when files are compared side-by-side. By default, the display is widened without changing its height. However, the user can set the above variable to indicate the name of a function to be called to widen the frame in which to display the buffers. See the on-line documentation for `ediff-make-wide-display-function` for details. It is also recommended to look into the source of the default function `ediff-make-wide-display`.

`ediff-use-last-dir`
: Controls the way Ediff presents the default directory when it prompts the user for files to compare. If `nil`, Ediff will use the default directory of the current buffer when it prompts the user for file names. Otherwise, it will use the directories it had previously used for files A, B, or C, respectively.

`ediff-no-emacs-help-in-control-buffer`
: If t, makes *C-h* behave like the *DEL* key, i.e., it will move you back to the previous difference rather than invoking help. This is useful when, in an xterm window or on a dumb terminal, the Backspace key is bound to *C-h* and is positioned more conveniently than the *DEL* key.

`ediff-toggle-read-only-function`
: Can be used to change the way Ediff toggles the read-only property in its buffers. By default, Ediff uses `toggle-read-only`. For files under version control, Ediff first tries to check the files out.

`ediff-keep-variants`
: Default is `t`, meaning that the buffers being compared or merged will be preserved when Ediff quits. Setting this to `nil` causes Ediff to offer the user a chance to delete these buffers (if they are not modified). Supplying a prefix argument to the quit command (`q`) temporarily reverses the meaning of this variable. This is convenient when the user prefers one of the behaviors most of the time, but occasionally needs the other behavior.

  Using `ediff-cleanup-hooks`, one can make Ediff delete the variants unconditionally (e.g., by making `ediff-janitor` into one of these hooks).

Ediff lets you toggle the way windows are split, so you can try different settings interactively. **Note**: if buffers A and B (and C, if applicable) are in different frames, windows are not split, regardless of the value `ediff-split-window-function`. Instead, other windows on these frames are deleted and Ediff starts displaying these buffers using these frames, one file per frame. You can switch to the one-frame mode by hiding one of the buffers A/B/C.

Note that if Ediff detects that the two buffers it compares are residing in separate frames, it assumes that the user wants them to be so displayed and stops splitting windows. Instead, it will arrange each buffer to occupy its own frame.

The user can also swap the windows where buffers are displayed by typing `~`.

7.13 Notes on Heavy-duty Customization

Some users need to customize Ediff in rather sophisticated ways, which requires different defaults for different kinds of files (e.g., SGML, etc.). Ediff supports this kind of customization is several ways. First, most customization variables are buffer-local. Those that aren't are usually accessible from within Ediff Control Panel, so one can make them local to the panel by calling make-local-variable from within `ediff-startup-hooks`. Second, there is now a new optional (6-th) argument to `ediff-setup`, which has the form `( (var-name-1 . val-1) (var-name-2 . val-2) ...)`. The function `ediff-setup` will set the variables on the list to the respective values in the Ediff control buffer. This is an easy way to throw in custom variables (which usually should be buffer-local) that can then be tested in various hooks. Make sure the variable `ediff-job-name` and `ediff-word-mode` are set properly in this case, as some things in Ediff depend on this. Finally, if custom-tailored help messages are desired, Ediff has `ediff-brief-help-message-custom` and `ediff-long-help-message-custom`, which are local variables that can be set to a function that returns a string.

When customizing Ediff, some other variables are useful, although they are not user-definable. First, it should be kept in mind that most of the Ediff variables are local to the Ediff control buffer, so this buffer must be current at the time these variables are accessed. The control buffer is accessible via the variable `ediff-control-buffer`, which is also local to that buffer.

Other variables of interest are:

`ediff-buffer-A`
: The first of the data buffers being compared.

`ediff-buffer-B`
: The second of the data buffers being compared.

`ediff-buffer-C`
: In three-way comparisons, this is the third buffer being compared. In merging, this is the merge buffer. In two-way comparison, this variable is nil.

`ediff-window-A`
: The window displaying buffer A. If buffer A is not visible, this variable is nil or it may be a dead window.

`ediff-window-B`
: The window displaying buffer B.

`ediff-window-C`
: The window displaying buffer C, if any.

`ediff-control-frame`
: A dedicated frame displaying the control buffer, if it exists. It is non-nil only if Ediff uses the multiframe display, i.e., when the control buffer is in its own frame.

8 Credits

Ediff was written by Michael Kifer <kifer@cs.sunysb.edu>. It was inspired by emerge.el written by Dale R. Worley <drw@math.mit.edu>. An idea due to Boris Goldowsky <boris@cs.rochester.edu> made it possible to highlight fine differences in Ediff buffers. Alastair Burt <burt@dfki.uni-kl.de> ported Ediff to XEmacs, and Eric Freudenthal <freudent@jan.ultra.nyu.edu> made it work with VC.

Many people provided help with bug reports, patches, and advice. Without them, Ediff would not be nearly as useful as it is now. Here is a full list of contributors (I hope I didn't miss anyone):

Alastair Burt <burt@dfki.uni-kl.de>, Paul Bibilo
<peb@delcam.co.uk>, Kevin Broadey
<KevinB@bartley.demon.co.uk>, Harald Boegeholz
<hwb@machnix.mathematik.uni-stuttgart.de>, Jin S. Choi
<jin@atype.com>, Eric Eide <eeide@asylum.cs.utah.edu>,
Kevin Esler <esler@ch.hp.com>, Robert Estes
<estes@ece.ucdavis.edu>, Eric Freudenthal
<freudent@jan.ultra.nyu.edu>, Job Ganzevoort
<Job.Ganzevoort@cwi.nl>, Boris Goldowsky
<boris@cs.rochester.edu>, Allan Gottlieb
<gottlieb@allan.ultra.nyu.edu>, Xiaoli Huang <hxl@epic.com>,
Larry Gouge <larry@itginc.com>, Karl Heuer
<kwzh@gnu.ai.mit.edu>, <irvine@lks.csi.com>,
<jaffe@chipmunk.cita.utoronto.ca>, David Karr
<dkarr@nmo.gtegsc.com>, Norbert Kiesel
<norbert@i3.informatik.rwth-aachen.de>,
Leigh L Klotz <klotz@adoc.xerox.com>, Fritz Knabe
<Fritz.Knabe@ecrc.de>, Heinz Knutzen
<hk@informatik.uni-kiel.d400.de>,
Andrew Koenig <ark@research.att.com>,
Ken Laprade <laprade@dw3f.ess.harris.com>, Will C Lauer <wcl@cadre.com>,
Richard Levitte <levitte@e.kth.se>, Mike Long <mike.long@analog.com>,
Martin Maechler <maechler@stat.math.ethz.ch>, Simon Marshall
<Simon.Marshall@mail.esrin.esa.it>, Richard Mlynarik
<mly@adoc.xerox.com>, Chris Murphy <murphycm@sun.aston.ac.uk>,
Eyvind Ness <Eyvind.Ness@hrp.no>, Ray Nickson
<nickson@cs.uq.oz.au>, Paul Raines <raines@slac.stanford.edu>,
Benjamin Pierce <benjamin.pierce@cl.cam.ac.uk>,
Tibor Polgar <tlp00@spg.amdahl.com>, C.S. Roberson
<roberson@aur.alcatel.com>, Kevin Rodgers <kevin.rodgers@ihs.com>,
Sandy Rutherford <sandy@ibm550.sissa.it>, Heribert Schuetz
<schuetz@ecrc.de>, Andy Scott <ascott@pcocd2.intel.com>,
Axel Seibert <axel@tumbolia.ppp.informatik.uni-muenchen.de>,
Richard Stallman <rms@gnu.ai.mit.edu>, Richard Stanton
<stanton@haas.berkeley.edu>, Ake Stenhoff
<etxaksf@aom.ericsson.se>, Stig <stig@hackvan.com>, Peter Stout

<Peter_Stout@cs.cmu.edu>, Chuck Thompson <cthomp@cs.uiuc.edu>,
Raymond Toy <toy@rtp.ericsson.se>, Ilya Zakharevich
<ilya@math.ohio-state.edu>

Index

C

Comparing files and buffers 1

E

ediff 2
ediff-after-flag-eol 15
ediff-after-flag-mol 15
ediff-after-setup-control-frame-hooks 11
ediff-after-setup-windows-hooks 10
ediff-auto-refine 17
ediff-auto-refine-limit 17
ediff-before-flag-bol 15
ediff-before-flag-mol 15
ediff-before-setup-control-frame-hooks 11
ediff-before-setup-windows-hooks 10
ediff-brief-help-message-custom 23
ediff-buffers 2
ediff-buffers3 2
ediff-cleanup-hooks 10
ediff-combination-pattern 19
ediff-control-buffer 12
ediff-control-frame-parameters 12
ediff-control-frame-position-function 12
ediff-control-frame-upward-shift 13
ediff-current-diff-face-A 15
ediff-current-diff-face-B 15
ediff-current-diff-face-C 15
ediff-custom-diff-options 18
ediff-custom-diff-program 18
ediff-default-variant 18
ediff-diff-options 18
ediff-diff-program 17, 18
ediff-diff3-options 18
ediff-diff3-program 18
ediff-directories 2
ediff-directories3 2
ediff-directory-revisions 2
ediff-display-help-hooks 11
ediff-even-diff-face-A 15
ediff-even-diff-face-B 15
ediff-even-diff-face-C 15
ediff-files 2
ediff-files3 2
ediff-fine-diff-face-A 15
ediff-fine-diff-face-B 15
ediff-fine-diff-face-C 15
ediff-forward-word 17
ediff-forward-word-function 17
ediff-help-message 12
ediff-highlight-all-diffs 15
ediff-ignore-similar-regions 14
ediff-janitor 11
ediff-job-name 23
ediff-keep-variants 22
ediff-keymap-setup-hooks 10
ediff-load-hooks 10
ediff-long-help-message-custom 23
ediff-make-frame-position 12
ediff-make-wide-display-function 22
ediff-merge 3
ediff-merge-buffers 3
ediff-merge-buffers-with-ancestor 3
ediff-merge-directories 3
ediff-merge-directories-with-ancestor 3
ediff-merge-directory-revisions 2
ediff-merge-directory-revisions-with-ancestor 2
ediff-merge-files 3
ediff-merge-files-with-ancestor 3
ediff-merge-revisions 3
ediff-merge-revisions-with-ancestor 3
ediff-merge-split-window-function 22
ediff-merge-window-share 20
ediff-merge-with-ancestor 3
ediff-mode-hooks 11
ediff-narrow-control-frame-leftward-shift 13
ediff-no-emacs-help-in-control-buffer 22
ediff-odd-diff-face-A 15
ediff-odd-diff-face-B 15
ediff-odd-diff-face-C 15
ediff-patch-buffer 3
ediff-patch-file 3
ediff-patch-options 18
ediff-patch-program 18
ediff-prefer-iconified-control-frame 13
ediff-prefer-long-help-message 12, 13
ediff-prepare-buffer-hooks 11, 21
ediff-profile 6
ediff-quit-hooks 10
ediff-regions-linewise 2, 16
ediff-regions-wordwise 2, 16
ediff-registry-setup-hooks 11
ediff-revert-buffers-then-recompute-diffs 6
ediff-revision 2
ediff-revision-key 21
ediff-save-buffer 18
ediff-select-hooks 11
ediff-session-group-setup-hooks 11
ediff-setup 23
ediff-setup-windows 13
ediff-setup-windows-multiframe 13
ediff-setup-windows-plain 13
ediff-show-clashes-only 20
ediff-show-registry 6
ediff-split-window-function 22
ediff-start-narrowed 16
ediff-startup-hooks 11, 12, 23
ediff-suspend-hooks 10

ediff-toggle-multiframe 6, 13
ediff-toggle-read-only-function 22
ediff-unselect-hooks 11
ediff-use-last-dir 3, 22
ediff-version-control-package 21
ediff-wide-control-frame-rightward-shift . 13
ediff-window-setup-function 13
ediff-windows-linewise 2, 16
ediff-windows-wordwise 2, 16
ediff-word-1 17
ediff-word-2 17
ediff-word-3 17
ediff-word-4 17
ediff-word-mode 23
ediff3 2
edir-merge-revisions 2
edir-merge-revisions-with-ancestor 2
edir-revisions 2
edirs 2
edirs-merge 3
edirs-merge-with-ancestor 3
edirs3 2
epatch 3
epatch-buffer 3
eregistry 6

F

Finding differences 1

M

Merging files and buffers 1
mode-line.el 21
Multi-file patches 8

P

Patching files and buffers 1

R

rcs.el 21

S

split-window-horizontally 22
split-window-vertically 22

U

uniquify.el 21

V

vc.el 21

Table of Contents

1 Introduction **1**

2 Major Entry Points **2**

3 Commands **5**

4 Registry of Ediff Sessions **7**

5 Session Groups **8**

6 Remote and Compressed Files **9**

7 Customization **10**

7.1 Hooks 10
7.2 Quick Help 11
7.3 Window and Frame Configuration 12
7.4 Selective Browsing 14
7.5 Highlighting Difference Regions 15
7.6 Narrowing 16
7.7 Refinement of Difference Regions 17
7.8 Patch and Diff Programs 17
7.9 Merging and diff3 18
7.10 Support for Version Control 21
7.11 Customizing the Mode Line 21
7.12 Miscellaneous 22
7.13 Notes on Heavy-duty Customization 23

8 Credits **25**

Index **27**