Dired Extra Version 2

For The GNU Emacs 19 Directory Editor

Manual Revision: 2.52
1994/08/09 16:51:31

Lawrence R. Dodd
dodd@roebling.poly.edu

(Based on 'dired.texi' by Sebastian Kremer <sk@thp.uni-koeln.de>)

The file used to create this is called 'dired-x.texi', but the original work that was altered to make that file was called 'dired.texi' written by Sebastian Kremer.

1 Introduction

This documents the *extra* features for Dired Mode for GNU Emacs 19. It is derived from version 1.191 of Sebastian Kremer's '`dired-x.el`' and is GNU Emacs v19 compatible.

In adopting this '`dired-x.el`' to GNU Emacs v19 some material that has been incorporated into '`dired.el`' and '`dired-aux.el`' of the GNU Emacs 19 distribution has been removed and some material was modified for agreement with the functions in '`dired.el`' and '`dired-aux.el`'. For example, the code using `gmhist` history functions was replaced with code using the mini-buffer history now built into GNU Emacs 19. Finally, a few other features have been added and a few more functions have been bound to keys.

Please note that '`dired-x.el`' and this texinfo file '`dired-x.texi`' are bundled with GNU Emacs versions 19.23 and later.

1.1 Features

Some features provided by Dired Extra

1. Omitting of uninteresting files from dired listing.

   See Chapter 3 [Omitting Files in Dired], page 6.
2. Local variables for dired directories.

   See Chapter 4 [Local Variables], page 9.
3. Guessing shell commands in dired buffers.

   See Chapter 5 [Shell Command Guessing], page 10.
4. Running dired command in non-dired buffers.

   See Chapter 6 [Virtual Dired], page 12.
5. Finding a file mentioned in a buffer

   See Section 8.1 [Find File At Point], page 15.
6. Commands using file marking.

   See Chapter 7 [Advanced Mark Commands], page 13.

'`dired-x.el`' binds some functions to keys in Dired Mode (see [Key Index], page 22) and also binds *`C-x C-j`* and *`C-x 4 C-j`* *globally* to `dired-jump` (see Chapter 9 [Miscellaneous Commands], page 17). It may also bind *`C-x C-f`* and *`C-x 4 C-f`* to `dired-x-find-file` and `dired-x-find-file-other-window`, respectively (see Section 8.1 [Find File At Point], page 15).

1.2 Technical Details

When loaded this code *redefines* the following functions of GNU Emacs from '`dired.el`'

- `dired-clean-up-after-deletion`
- `dired-find-buffer-nocreate`
- `dired-initial-position`
- `dired-up-directory`

and the following functions from ‘dired-aux.el’

- dired-add-entry
- dired-read-shell-command

One drawback is that ‘dired-x.el’ will load ‘dired-aux.el’ as soon as dired is loaded. Thus, the advantage of separating out non-essential dired stuff into ‘dired-aux.el’ and only loading when necessary will be lost when ‘dired-x.el’ is used.

2 Installation

This manual describes the dired features provided by the file '`dired-x.el`'. To take advantage of these features, you must load the file and (optionally) set some variables.

In your '`.emacs`' file in your home directory, or in the system-wide initialization file '`default.el`' in the '`site-lisp`' directory, put

```
(add-hook 'dired-load-hook
          (function (lambda ()
                      (load "dired-x")
                      ;; Set dired-x global variables here.  For example:
                      ;; (setq dired-guess-shell-gnutar "gtar")
                      ;; (setq dired-x-hands-off-my-keys nil)
                      )))
(add-hook 'dired-mode-hook
          (function (lambda ()
                      ;; Set dired-x buffer-local variables here.  For example:
                      ;; (setq dired-omit-files-p t)
                      )))
```

This will load '`dired-x.el`' when dired is first invoked (for example, when you first do *C-x d*).

2.1 Optional Installation Dired Jump

In order to have `dired-jump` and `dired-jump-other-window` (see Chapter 9 [Miscellaneous Commands], page 17) work *before* `dired` and `dired-x` have been properly loaded the user should set-up an autoload for these functions. In your '`.emacs`' file put

```
;;; Autoload 'dired-jump' and 'dired-jump-other-window'.
;;; We autoload from FILE dired.el.  This will then load dired-x.el
;;; and hence define 'dired-jump' and 'dired-jump-other-window'.
(define-key global-map "\C-x\C-j" 'dired-jump)
(define-key global-map "\C-x4\C-j" 'dired-jump-other-window)

(autoload (quote dired-jump) "dired" "\
Jump to dired buffer corresponding to current buffer.
If in a file, dired the current directory and move to file's line.
If in dired already, pop up a level and goto old directory's line.
In case the proper dired file line cannot be found, refresh the dired
buffer and try again." t nil)

(autoload (quote dired-jump-other-window) "dired" "\
Like \\[dired-jump] (dired-jump) but in other window." t nil)
```

Note that in recent releases of GNU Emacs 19 (i.e., 19.25 or later) the file '`../lisp/loaddefs.el`' of the Emacs distribution already contains the proper auto-loading for `dired-jump` so you need only put

```
(define-key global-map "\C-x\C-j" 'dired-jump)
```

in your '`.emacs`' file in order to have *C-x C-j* work before `dired` is loaded.

2.2 Optional Installation File At Point

If you choose to have 'dired-x.el' bind dired-x-find-file over find-file (see Section 8.1 [Find File At Point], page 15), then you will need to set dired-x-hands-off-my-keys and make a call to the function dired-x-bind-find-file in the dired-load-hook:

```
(add-hook 'dired-load-hook
          (function (lambda ()
                      (load "dired-x")
                      ;; Bind dired-x-find-file.
                      (setq dired-x-hands-off-my-keys nil)
                      ;; Make sure our binding preference is invoked.
                      (dired-x-bind-find-file)
                      )))
```

Alternatively, you can set the variable *before* 'dired-x.el' is loaded

```
(add-hook 'dired-load-hook
          (function (lambda ()
                      ;; Bind dired-x-find-file.
                      (setq dired-x-hands-off-my-keys nil)
                      (load "dired-x")
                      )))
```

2.3 Special Notes

If 'dired-x.el' was *not* bundled with the version of GNU Emacs installed at your site (i.e., not in the default '../lisp' directory) then you must put the file 'dired-x.el' in a directory known to GNU Emacs. Examine the variable load-path for a list of these directories. If you wish to add a new directory on this list of directories use something like this in your '.emacs' file

```
;;; LOAD PATH
(setq load-path (append
                 load-path ; default at top
                 (list
                  "/the/directory/where/you/put/dired-x")))
```

If you wish to put the new directory at the head of the list (where it will be found first) then you should use instead

```
;;; LOAD PATH
(setq load-path (append
                 (list
                  "/the/directory/where/you/put/dired-x")
                 load-path)) ; default at bottom
```

You must also byte compile the file (for example, hitting *B* in dired-mode). When byte-compiling 'dired-x.el' you may get messages about functions vm-visit-folder, Man-notify-when-ready, and reporter-submit-bug-report not being defined. These are warnings and should be ignored.

CAUTION: If you are using a version of GNU Emacs earlier than 19.20 than you may have to edit 'dired.el'. The copy of 'dired.el' in GNU Emacs versions earlier than 19.20

incorrectly had the call to `run-hooks` *before* the call to `provide`. In such a case, it is possible that byte-compiling and/or loading dired can cause an infinite loop. To prevent this, make sure the line of code

```
(run-hooks 'dired-load-hook)
```

is the *last* executable line in the file '`dired.el`'. That is, make sure it comes *after* the line

```
(provide 'dired)
```

3 Omitting Files in Dired

Omitting a file means removing it from the directory listing. Omitting is useful for keeping Dired buffers free of "uninteresting" files (for instance, auto-save, auxiliary, backup, and revision control files) so that the user can concentrate on the interesting files. Like hidden files, omitted files are never seen by Dired. Omitting differs from hiding in several respects:

- Omitting works on individual files, not on directories; an entire directory cannot be omitted (though each of its files could be).
- Omitting is wholesale; if omitting is turned on for a dired buffer, then all uninteresting files listed in that buffer are omitted. The user does not omit (or unomit) files one at a time.
- Omitting can be automatic; uninteresting file lines in the buffer can be removed before the user ever sees them.
- Marked files are never omitted.

M-o (`dired-omit-toggle`) Toggle between displaying and omitting "uninteresting" files. With a prefix argument, don't toggle and just mark the files, but don't actually omit them.

In order to make Dired Omit work you first need to load '`dired-x.el`' inside `dired-load-hook` (see Chapter 2 [Installation], page 3) and then set `dired-omit-files-p` in some way (see Section 3.1 [Omitting Variables], page 6).

3.1 Omitting Variables

The following variables can be used to customize omitting.

`dired-omit-files-p`

Default: `nil`

If non-`nil`, "uninteresting" files are not listed. Uninteresting files are those whose filenames match regexp `dired-omit-files`, plus those ending with extensions in `dired-omit-extensions`. *M-o* (`dired-omit-toggle`) toggles its value, which is buffer-local. Put

```
(setq dired-omit-files-p t)
```

inside your `dired-mode-hook` to have omitting initially turned on in *every* Dired buffer (see Chapter 2 [Installation], page 3). You can then use *M-o* to unomit in that buffer.

To enable omitting automatically only in certain directories one can use Dired Local Variables and put

```
Local Variables:
dired-omit-files-p: t
End:
```

into a file '`.dired`' (the default value of `dired-local-variables-file`) in that directory (see Chapter 4 [Local Variables], page 9).

`dired-omit-here-always`
This is an interactive function that creates a local variables file exactly like the example above (if it does not already exist) in the file `dired-local-variables-file` in the current directory and then refreshes the directory listing (see Chapter 4 [Local Variables], page 9).

`dired-omit-files`
Default: `"^#\\|\\.$"`

Filenames matching this buffer-local regexp will not be displayed. This only has effect when `dired-omit-files-p` is t.

The default value omits the special directories '`.`' and '`..`' and autosave files (plus other files ending in ".") (see Section 3.2 [Omitting Examples], page 7).

`dired-omit-extensions`
Default: The elements of `completion-ignored-extensions` (as defined in the file '`loaddefs.el`' of the GNU Emacs distribution), `dired-latex-unclean-extensions`, `dired-bibtex-unclean-extensions` and `dired-texinfo-unclean-extensions`.

If non-`nil`, a list of extensions (strings) to omit from Dired listings. Its format is the same as that of `completion-ignored-extensions`.

`dired-omit-localp`
Default: `'no-dir`

The *localp* argument `dired-omit-expunge` passes to `dired-get-filename`. If it is `'no-dir`, omitting is much faster, but you can only match against the non-directory part of the filename. Set it to `nil` if you need to match the whole pathname or `t` to match the pathname relative to the buffer's top-level directory.

`dired-omit-marker-char`
Default: *C-o*

Temporary marker used by by Dired to implement omitting. Should never be used as marker by the user or other packages. There is one exception to this rule: by doing

```
(setq dired-mark-keys "\C-o")
;; i.e., the value of dired-omit-marker-char
;; (which is not defined yet)
```

anywhere in your '`~/.emacs`', you will bind the *C-o* key to insert a *C-o* marker, thus causing these files to be omitted in addition to the usually omitted files. Unfortunately the files you omitted manually this way will show up again after reverting the buffer, unlike the others.

3.2 Examples of Omitting Various File Types

- If you wish to avoid seeing RCS files and the RCS directory, then put

```
(setq dired-omit-files
      (concat dired-omit-files "\\|^RCS$\\|,v$"))
```

in the `dired-load-hook` (see Chapter 2 [Installation], page 3). This assumes `dired-omit-localp` has its default value of `'no-dir` to make the `^`-anchored matches work. As a slower alternative, with `dired-omit-localp` set to `nil`, you can use `/` instead of `^` in the regexp.

- If you use tib, the bibliography program for use with TeX and LaTeX, you might want to omit the '`INDEX`' and the '`-t.tex`' files, then put

```
(setq dired-omit-files
      (concat dired-omit-files "\\|^INDEX$\\|-t\\.tex$"))
```

  in the `dired-load-hook` (see Chapter 2 [Installation], page 3).

- If you do not wish to see '`dot`' files (files starting with a '`.`'), then put

```
(setq dired-omit-files
      (concat dired-omit-files "\\|^\\..+$"))
```

  in the `dired-load-hook` (see Chapter 2 [Installation], page 3).

3.3 Some Technical Details of Omitting

Loading '`dired-x.el`' will install Dired Omit by putting `dired-omit-expunge` on your `dired-after-readin-hook`, and will call `dired-extra-startup`, which in turn calls `dired-omit-startup` in your `dired-mode-hook`.

4 Local Variables for Dired Directories

When Dired visits a directory, it looks for a file whose name is the value of variable `dired-local-variables-file` (default: '`.dired`'). If such a file is found, Dired will temporarily insert it into the Dired buffer and run `hack-local-variables`.

For example, if the user puts

```
Local Variables:
dired-actual-switches: "-lat"
dired-omit-files-p: t
End:
```

into a file called '`.dired`' in a directory then when that directory is viewed it will be

1. sorted by date
2. omitted automatically

You can set `dired-local-variables-file` to `nil` to suppress this. The value of `dired-enable-local-variables` controls if and how these local variables are read. This variable exists so that if may override the default value of `enable-local-variables`.

Please see the GNU Emacs Manual to learn more about local variables. See section "Local Variables in Files" in *The GNU Emacs Manual*.

The following variables affect Dired Local Variables

`dired-local-variables-file`

Default: `".dired"`

If non-`nil`, filename for local variables for Dired. If Dired finds a file with that name in the current directory, it will temporarily insert it into the dired buffer and run 'hack-local-variables'.

`dired-enable-local-variables`

Default: `t`

Controls use of local-variables lists in dired. The value can be `t`, `nil`, or something else. A value of `t` means local-variables lists are obeyed in the `dired-local-variables-file`; `nil` means they are ignored; anything else means query. This variable temporarily overrides the value of `enable-local-variables` when the Dired Local Variables are hacked.

5 Shell Command Guessing

Based upon the name of a filename, Dired tries to guess what shell command you might want to apply to it. For example, if you have point on a file named '`foo.tar`' and you press *!*, Dired will guess you want to '`tar xvf`' it and suggest that as the default shell command.

The default will be mentioned in brackets and you can type *M-p* to get the default into the minibuffer so that you can edit it, e.g., changing '`tar xvf`' to '`tar tvf`'. If there are several commands for a given file, e.g., '`xtex`' and '`dvips`' for a '`.dvi`' file, you can type *M-p* several times to see each of the matching commands.

Dired only tries to guess a command for a single file, never for a list of marked files.

`dired-guess-shell-alist-default`
: Predefined rules for shell commands. Set this to `nil` to turn guessing off. The elements of `dired-guess-shell-alist-user` (defined by the user) will override these rules.

`dired-guess-shell-alist-user`
: If non-`nil`, a user-defined alist of file regexps and their suggested commands. These rules take precedence over the predefined rules in the variable `dired-guess-shell-alist-default` (to which they are prepended) when `dired-do-shell-command` is run).

  Each element of the alist looks like

  ```
  (regexp command...)
  ```

  where each *command* can either be a string or a lisp expression that evaluates to a string. If several *COMMAND*s are given, all will temporarily be pushed on the history.

  You can set this variable in your '`~/.emacs`'. For example, to add rules for '`.foo`' and '`.bar`' file extensions, write

  ```
  (setq dired-guess-shell-alist-user
        (list
         (list "\\.foo$" "foo-command");; fixed rule
         ;; possibly more rules...
         (list "\\.bar$";; rule with condition test
               '(if condition
                    "bar-command-1"
                  "bar-command-2"))))
  ```

  This will override any predefined rules for the same extensions.

`dired-guess-shell-gnutar`
: Default: `nil`

  If non-`nil`, name of the GNU tar executable (e.g., '`"tar"`' or '`"gnutar"`'). GNU tar's '`z`' switch is used for compressed tar files. If you don't have GNU tar, set this to `nil`: a pipe using '`zcat`' is then used.

`dired-guess-shell-gzip-quiet`
: Default: `t`

  A non-`nil` value means that `-q` is passed to gzip overriding a verbose GNU zip's '`GZIP`' environment variable.

`dired-guess-shell-znew-switches` nil
: Default: `nil`

  A string of switches passed to GNU zip's '`znew`'. An example is '`"-K"`' which will make '`znew`' keep a .Z file when it is smaller than the .gz file.

`dired-shell-command-history` nil
: History list for commands that read dired-shell commands.

6 Virtual Dired

Using *Virtual Dired* means putting a buffer with Dired-like contents in Dired mode. The files described by the buffer contents need not actually exist. This is useful if you want to peruse an '`ls -lR`' output file, for example one you got from an FTP server. You can use all motion commands usually available in Dired. You can also use it to save a Dired buffer in a file and resume it in a later session.

Type *`M-x dired-virtual`* to put the current buffer into virtual Dired mode. You will be prompted for the top level directory of this buffer, with a default value guessed from the buffer contents. To convert the virtual to a real Dired buffer again, type *`g`* (which calls `dired-virtual-revert`) in the virtual Dired buffer and answer '`y`'. You don't have to do this, though: you can relist single subdirectories using *`l`* (`dired-do-redisplay`) on the subdirectory headerline, leaving the buffer in virtual Dired mode all the time.

The function '`dired-virtual-mode`' is specially designed to turn on virtual Dired mode from the `auto-mode-alist`. To edit all '`*.dired`' files automatically in virtual Dired mode, put this into your '`~/.emacs`':

```
(setq auto-mode-alist (cons '("[^/]\\.dired$" . dired-virtual-mode)
                            auto-mode-alist))
```

The regexp is a bit more complicated than usual to exclude ".dired" local variable files.

7 Advanced Mark Commands

F
: (`dired-do-find-marked-files`) Find all marked files at once displaying simultaneously. If optional NOSELECT is non-`nil` then just find the files but do not select. If you want to keep the dired buffer displayed, type *C-x 2* first. If you want just the marked files displayed and nothing else, type *C-x 1* first.

  The current window is split across all files marked, as evenly as possible. Remaining lines go to the bottom-most window. The number of files that can be displayed this way is restricted by the height of the current window and the variable `window-min-height`.

`dired-mark-extension`
: Mark all files with a certain extension for use in later commands. A '`.`' is not automatically prepended to the string entered.

  When called from lisp, *extension* may also be a list of extensions and an optional argument *marker-char* specifies the marker used.

`dired-flag-extension`
: Flag all files with a certain extension for deletion. A '`.`' is *not* automatically prepended to the string entered.

7.1 Advanced Cleaning Functions

`dired-clean-patch`
: Flag dispensable files created by the '`patch`' program for deletion. See variable `dired-patch-unclean-extensions`.

`dired-clean-tex`
: Flag dispensable files created by TeX, LaTeX, and '`texinfo`' for deletion. See the following variables (see Section 7.2 [Advanced Cleaning Variables], page 13):

  - `dired-tex-unclean-extensions`
  - `dired-texinfo-unclean-extensions`
  - `dired-latex-unclean-extensions`
  - `dired-bibtex-unclean-extensions`

`dired-very-clean-tex`
: Flag dispensable files created by TeX, LaTeX, '`texinfo`', and ".dvi" files for deletion.

7.2 Advanced Cleaning Variables

Variables used by the above cleaning commands (and in the default value for variable `dired-omit-extensions`, see Section 3.1 [Omitting Variables], page 6)

`dired-patch-unclean-extensions`
: Default: `'(".rej" ".orig")`

  List of extensions of dispensable files created by the '`patch`' program.

`dired-tex-unclean-extensions`
: Default: `'(".toc" ".log" ".aux")`

  List of extensions of dispensable files created by TeX.

`dired-texinfo-unclean-extensions`
: Default: `'(".cp" ".cps" ".fn" ".fns" ".ky" ".kys" ".pg" ".pgs" ".tp" ".tps" ".vr" ".vrs")`

  List of extensions of dispensable files created by '`texinfo`'.

`dired-latex-unclean-extensions`
: Default: `'(".idx" ".lof" ".lot" ".glo")`

  List of extensions of dispensable files created by LaTeX.

`dired-bibtex-unclean-extensions`
: Default: `'(".blg" ".bbl")`

  List of extensions of dispensable files created by BibTeX.

7.3 Special Marking Function

`M-(`
: (`dired-mark-sexp`) Mark files for which *predicate* returns non-`nil`. With a prefix argument, unflag those files instead.

  The *predicate* is a lisp expression that can refer to the following symbols:

  `inode` — [*integer*] the inode of the file (only for '`ls -i`' output)

  `s` — [*integer*] the size of the file for '`ls -s`' output (usually in blocks or, with '`-k`', in KBytes)

  `mode` — [*string*] file permission bits, e.g., '`"-rw-r--r--"`'

  `nlink` — [*integer*] number of links to file

  `uid` — [*string*] owner

  `gid` — [*string*] group (If the gid is not displayed by '`ls`', this will still be set (to the same as uid))

  `size` — [*integer*] file size in bytes

  `time` — [*string*] the time that '`ls`' displays, e.g., '`"Feb 12 14:17"`'

  `name` — [*string*] the name of the file

  `sym` — [*string*] if file is a symbolic link, the linked-to name, else '`""`'

  For example, use

  ```
  (equal 0 size)
  ```

  to mark all zero length files.

  To find out all not yet compiled Emacs lisp files in a directory, dired all '`.el`' files in the lisp directory using the wildcard '`*.el`'. Then use *`M-(`* with

  ```
  (not (file-exists-p (concat name "c")))
  ```

  to mark all '`.el`' files without a corresponding '`.elc`' file.

8 Multiple Dired Directories and Non-Dired Commands

An Emacs buffer can have but one working directory, stored in the buffer-local variable `default-directory`. A Dired buffer may have several subdirectories inserted, but still has but one working directory: that of the top level Dired directory in that buffer. For some commands it is appropriate that they use the current Dired directory instead of `default-directory`, e.g., `find-file` and `compile`.

A general mechanism is provided for special handling of the working directory in special major modes:

`default-directory-alist`
: Default: `((dired-mode . (dired-current-directory)))`

  Alist of major modes and their opinion on `default-directory`, as a lisp expression to evaluate. A resulting value of `nil` is ignored in favor of `default-directory`.

`default-directory`
: Function with usage like variable `default-directory`, but knows about the special cases in variable `default-directory-alist`.

8.1 Find File At Point

'`dired-x`' provides a method of visiting or editing a file mentioned in the buffer you are viewing (e.g., a mail buffer, a news article, a README file, etc.) or to test if that file exists. You can then modify this in the minibuffer after snatching the filename.

When installed '`dired-x`' will substitute `dired-x-find-file` for `find-file` (normally bound to *C-x C-f*) and `dired-x-find-file-other-window` for `find-file-other-window` (normally bound to *C-x 4 C-f*).

In order to use this feature, you will need to set `dired-x-hands-off-my-keys` to `nil` inside `dired-load-hook` (see Section 2.2 [Optional Installation File At Point], page 4).

`dired-x-find-file`
: `dired-x-find-file` behaves exactly like `find-file` (normally bound to *C-x C-f*) unless a prefix argument is passed to the function in which case it will use the filename at point as a guess for the file to visit.

  For example, if the buffer you were reading contained the words

  ```
  Available via anonymous ftp in

     /roebling.poly.edu:/pub/lisp/crypt++.el.gz
  ```

  then you could move your cursor to the line containing the ftp address and type *C-u C-x C-f* (the *C-u* is a universal argument). The minibuffer would read

  ```
  Find file: /roebling.poly.edu:/pub/lisp/crypt++.el.gz
  ```

  with the point after the last `/`. If you hit return emacs will visit the file at that address. This also works with files on your own computer.

`dired-x-find-file-other-window`
: `dired-x-find-file-other-window` behaves exactly like `find-file-other-window` (normally bound to *C-x 4 C-f*) unless a prefix argument is used. See `dired-x-find-file` for more information.

`dired-x-hands-off-my-keys`
: If set to `t`, then it means that '`dired-x`' should *not* bind `dired-x-find-file` over `find-file` on keyboard. Similarly, it should not bind `dired-x-find-file-other-window` over `find-file-other-window`. If you change this variable after '`dired-x.el`' is loaded then do *`M-x dired-x-bind-find-file`*. The default value of this variable is *`t`*; by default, the binding is not done. See See Section 2.2 [Optional Installation File At Point], page 4.

`dired-x-bind-find-file`
: A function, which can be called interactively or in your '`~/.emacs`' file, that uses the value of `dired-x-hands-off-my-keys` to determine if `dired-x-find-file` should be bound over `find-file` and `dired-x-find-file-other-window` bound over `find-file-other-window`. See See Section 2.2 [Optional Installation File At Point], page 4.

9 Miscellaneous Commands

Miscellaneous features not fitting anywhere else:

`dired-find-subdir`
: Default: `nil`

  If non-`nil`, Dired does not make a new buffer for a directory if it can be found (perhaps as subdirectory) in some existing Dired buffer.

  If there are several Dired buffers for a directory, the most recently used is chosen.

  Dired avoids switching to the current buffer, so that if you have a normal and a wildcard buffer for the same directory, *C-x d RET* will toggle between those two.

M-g
: (`dired-goto-file`) Goto file line of a file (or directory).

M-G
: (`dired-goto-subdir`) Goto headerline of an inserted directory. This commands reads its argument with completion over the names of the inserted subdirectories.

w
: (`dired-copy-filename-as-kill`) The *w* command puts the names of the marked (or next N) files into the kill ring, as if you had killed them with *C-w*. With a zero prefix argument N=0, use the complete pathname of each file. With a raw (just *C-u*) prefix argument, use the relative pathname of each marked file. As a special case, if no prefix argument is given and point is on a directory headerline, it gives you the name of that directory, without looking for marked files.

  The list of names is also stored onto the variable `dired-marked-files` for use, e.g., in the *M-:* (`eval-expression`) command.

  As this command also displays what was pushed onto the kill ring you can use it to display the list of currently marked files in the echo area (unless you happen to be on a subdirectory headerline).

  You can then feed the file name to other Emacs commands with *C-y*. For example, say you want to rename a long filename to a slightly different name. First type *w* to push the old name onto the kill ring. Then type *R* to rename it and use *C-y* inside *R*'s minibuffer prompt to insert the old name at a convenient place.

T
: (`dired-do-toggle`) Toggle marks. That is, currently marked files become unmarked and vice versa. Files marked with other flags (such as 'D') are not affected. The special directories '.' and '..' are never toggled.

`dired-smart-shell-command`
: Like function `shell-command`, but in the current Dired directory. Bound to *M-!* in Dired buffers.

`dired-jump`
: Bound to *C-x C-j*. Jump back to dired: If in a file, dired the current directory and move to file's line. If in Dired already, pop up a level and goto old directory's line. In case the proper Dired file line cannot be found, refresh the Dired buffer and try again.

`dired-jump-other-window`
: Bound to *C-x 4 C-j*. Like `dired-jump`, but to other window.

  These functions can be autoloaded so they work even though '`dired-x.el`' has not been loaded yet (see Section 2.1 [Optional Installation Dired Jump], page 3).

  If the variable `dired-bind-jump` is `nil`, `dired-jump` will not be bound to *C-x C-j* and `dired-jump-other-window` will not be bound to *C-x 4 C-j*.

`dired-vm`
: Bound to *V* if `dired-bind-vm` is t. Run VM on this file (assumed to be a UNIX mail folder).

  If you give this command a prefix argument, it will visit the folder read-only. This only works in VM~5, not VM~4.

  If the variable `dired-vm-read-only-folders` is t, `dired-vm` will visit all folders read-only. If it is neither `nil` nor `t`, e.g., the symbol `'if-file-read-only`, only files not writable by you are visited read-only. This is the recommended value if you run VM 5.

  If the variable `dired-bind-vm` is t, `dired-vm` will be bound to *V*. Otherwise, `dired-bind-rmail` will be bound.

`dired-rmail`
: Bound to *V* if `dired-bind-vm` is `nil`. Run Rmail on this file (assumed to be mail folder in Rmail/BABYL format).

`dired-info`
: Bound to *I*. Run Info on this file (assumed to be a file in Info format).

  If the variable `dired-bind-info` is `nil`, `dired-info` will not be bound to I.

`dired-man`
: Bound to *N*. Run man on this file (assumed to be a file in nroff format).

  If the variable `dired-bind-man` is `nil`, `dired-man` will not be bound to N.

`dired-do-relative-symlink`
: Bound to *Y*. Relative symlink all marked (or next ARG) files into a directory, or make a relative symbolic link to the current file. This creates relative symbolic links like

  foo -> ../bar/foo

  not absolute ones like

  foo -> /ugly/path/that/may/change/any/day/bar/foo

`dired-do-relative-symlink-regexp`
: Bound to *%Y*. Relative symlink all marked files containing REGEXP to NEW-NAME. See functions 'dired-do-rename-regexp' and 'dired-do-relsymlink' for more info.

10 Bugs

If you encounter a bug in this package, wish to suggest an enhancement, or want to make a smart remark, then type

```
M-x dired-x-submit-report
```

to set up an outgoing mail buffer, with the proper address to the '`dired-x.el`' maintainer automatically inserted in the '`To:`' field. This command also inserts information that the Dired X maintainer can use to recreate your exact setup, making it easier to verify your bug or social maladjustment.

Lawrence R. Dodd <dodd@roebling.poly.edu>

Concept Index

Function Index

Key Index

Variable Index

Table of Contents