- NAME
- SYNOPSIS
- DESCRIPTION
- RESOURCES
- OPTIONS
- CONCEPTS
  - TAGS
  - PREVIEWS
    - w3m
    - iTerm2
    - urxvt
    - urxvt-full
  - SELECTION
  - MACROS
  - BOOKMARKS
  - RIFLE
  - FLAGS
  - PLUGINS
- KEY BINDINGS
  - MAIN BINDINGS
  - READLINE-LIKE BINDINGS IN THE CONSOLE
- MOUSE BUTTONS
- SETTINGS
- COMMANDS
- FILES
  - CONFIGURATION
  - STORAGE
- ENVIRONMENT
- EXAMPLES
- LICENSE
- LINKS
- SEE ALSO
- BUGS

NAME

ranger - visual file manager

SYNOPSIS

ranger [**--version**] [**--help**] [**--debug**] [**--clean**]
[**--confdir**=*directory*] [**--copy-config**=*which*] [**--choosefile**=*target*]
[**--choosefiles**=*target*] [**--choosedir**=*target*] [**--selectfile**=*filepath*]
[**--list-unused-keys**] [**--list-tagged-files**=*tag*] [**--profile**]
[**--cmd**=*command*] [*path*]

DESCRIPTION

ranger is a console file manager with VI key bindings.

RESOUR

ranger.

for a list of key bindings, commands

The *README* contains install instructio

The file *HACKING* contains guidelines for code modification.

The directory *doc/* contains configuration files. They are usually installed to *sr/sha* and can be obtained w th ranger's --copy-config option

The directory *exa* contains reference implementations for ranger plugins, sample configuration files and some programs for integrating ranger w th other software. They are usually installed t

.

The man page of ri) describes the functions of the file oper

The section *LINKS* of this man page contains further resources.

OPTIONS

-d, **--debug**

Activate the debug mode: Whenever an error occurs, ranger will exit and print a full traceback. The default behavior is to merely print the name of the exception in the statusbar/log and try to keep running.

-c, **--clean**

Activate the clean mode: ranger will not access or create any configuration files nor will it leave any traces on your system. This is useful when your configuration is broken, when you want to avoid clutter, etc.

-r *dir*, **--confdir**=*dir*

Change the configuration directory of ranger from ~/.config /ranger to "dir".

--copy-config=*file*

Create copies of the default configuration files in your local configuration directory. Existing ones will not be overwritten. Possible values: *all*, *commands*, *commands_full*, *rc*, *rifle*, *scope*.

Note: You may want to disable loading of the global configuration files by exporting *RANGER_LOAD_DEFAULT_RC=FALSE* in your environment. See also: **FILES**, **ENVIRONMENT**

--copy-config=**commands** will copy only a small sample configuration file with a thoroughly commented example. It is recommended to keep this file tidy to avoid getting defunct commands on ranger upgrades. The full default commands.py can be copied with --copy-config=**commands_full**, but that file will be ignored by ranger and serves only as a reference for making your own commands.

--choosefile=*targetfile*

Allows you to pick a file with ranger. This changes the behavior so that when you open a file, ranger will exit and write the absolute path of that file into *targetfile*.

--choosefiles=*targetfile*

Allows you to pick multiple files with ranger. This changes the behavior so that when you open a file, ranger will exit and write the absolute paths of all selected files into *targetfile*, adding one newline after each filename.

--choosedir=*targetfile*

Allows you to pick a directory with ranger. When you exit ranger, it will write the last visited directory into *targetfile*.

--selectfile=*targetfile*

Open ranger with *targetfile* selected.

--list-unused-keys

List common keys which are not bound to any action in the "browser" context. This list is not complete, you can bind any key that is supported by curses: use the key code returned by `getch()`.

--list-tagged-files=*tag*

List all files which are tagged with the given tag. Note: Tags are single characters. The default tag is "*"

--profile

Print statistics of CPU usage on exit.

--cmd=*command*

Execute the command after the configuration has been read. Use this option multiple times to run multiple commands.

--version

Print the version and exit.

-h, **--help**

Print a list of options and exit.

CONCEPTS

This part explains how certain parts of ranger work and how they can be used efficiently.

TAGS

Tags are single characters which are displayed left of a filename. You can use tags however you want. Press "t" to toggle tags and "ut" to remove any tags of the selection. The default tag is an Asterisk ("*"), but you can use any tag by typing *"<tagname>*.

PREVIEWS

By default, only text files are previewed, but you can enable external preview scripts by setting the option `use_preview_script` and `preview_files` to true.

This default script is *~/.config/ranger/scope.sh*. It contains more documentation and calls to the programs *lynx* and *elinks* for html, *highlight* for text/code, *img2txt* for images, *atool* for archives, *pdftotext* for PDFs and *mediainfo* for video and audio files.

Install these programs (just the ones you need) and scope.sh will automatically use them.

Independently of the preview script, there is a feature to preview images by drawing them directly into the terminal. To enable this feature, set the option `preview_images` to true and enable one of the image

preview modes:

w3m

This does not work over ssh, requires certain terminals (tested on "xterm" and "urxvt") and is incompatible with tmux, although it works with screen.

To enable this feature, install the program "w3m" and set the option `preview_images_method` to w3m.

iTerm2

This only works in iTerm2 compiled with image preview support, but works over ssh.

To enable this feature, set the option `preview_images_method` to iterm2.

urxvt

This only works in urxvt compiled with pixbuf support. Does not work over ssh.

Essentially this mode sets an image as a terminal background temporarily, so it will break any previously set image background.

To enable this feature, set the option `preview_images_method` to urxvt.

urxvt-full

The same as urxvt but utilizing not only the preview pane but the whole terminal window.

To enable this feature, set the option `preview_images_method` to urxvt-full.

SELECTION

The *selection* is defined as "All marked files IF THERE ARE ANY, otherwise the current file." Be aware of this when using the :delete

command, which deletes all files in the selection.

You can mark files by pressing <Space>, v, etc. A yellow **Mrk** symbol at the bottom right indicates that there are marked files in this directory.

MACROS

Macros can be used in commands to abbreviate things.

```
%f   the highlighted file
%d   the path of the current directory
%s   the selected files in the current directory
%t   all tagged files in the current directory
%c   the full paths of the currently copied/cut files
%p   the full paths of selected files
```

The macros %f, %d, %p, and %s also have upper case variants, %F, %D, %P, and %S, which refer to the next tab. To refer to specific tabs, add a number in between. (%7s = selection of the seventh tab.)

%c is the only macro which ranges out of the current directory. So you may "abuse" the copying function for other purposes, like diffing two files which are in different directories:

```
Yank the file A (type yy), move to the file B, then type
@diff %c %f
```

Macros for file paths are generally shell-escaped so they can be used in the `shell` command.

Additionally, if you create a key binding that uses <any>, a special statement which accepts any key, then the macro %any (or %any0, %any1, %any2, ...) can be used in the command to get the key that was pressed.

The macro %rangerdir expands to the directory of ranger's python library, you can use it for something like this command: alias show_commands shell less %rangerdir/config/commands.py

%confdir expands to the directory given by **--confdir**.

The macro %space expands to a space character. You can use it to

add spaces to the end of a command when needed, while preventing editors to strip spaces off the end of the line automatically.

To write a literal %, you need to escape it by writing %%.

BOOKMARKS

Type **m<key>** to bookmark the current directory. You can re-enter this directory by typing **`<key>**. <key> can be any letter or digit. Unlike vim, both lowercase and uppercase bookmarks are persistent.

Each time you jump to a bookmark, the special bookmark at key ` will be set to the last directory. So typing "``" gets you back to where you were before.

Bookmarks are selectable when tabbing in the :cd command.

Note: The bookmarks ' (Apostrophe) and ` (Backtick) are the same.

RIFLE

Rifle is the file opener of ranger. It can be used as a standalone program or a python module. It is located at *ranger/ext/rifle.py*. In contrast to other, more simple file openers, rifle can automatically find installed programs so it can be used effectively out of the box on a variety of systems.

It's configured in *rifle.conf* through a list of conditions and commands. For each line the conditions are checked and if they are met, the respective command is taken into consideration. By default, simply the first matching rule is used. In ranger, you can list and choose rules by typing "r" or simply by typing "<rulenumber> <enter>". If you use rifle standalone, you can list all rules with the "-l" option and pick a rule with "-p <number>".

The rules, along with further documentation, are contained in *ranger/config/rifle.conf*.

FLAGS

Flags give you a way to modify the behavior of the spawned process. They are used in the commands `:open_with` (key "r") and `:shell` (key "!").

```
f   Fork the process.  (Run in background)
c   Run the current file only, instead of the selection
r   Run application with root privilege (requires sudo)
t   Run application in a new terminal window
```

There are some additional flags that can currently be used only in the `shell` command: (for example `:shell -w df`)

```
p   Redirect output to the pager
s   Silent mode.  Output will be discarded.
w   Wait for an Enter-press when the process is done
```

By default, all the flags are off unless specified otherwise in the *rifle.conf* configuration file. You can specify as many flags as you want. An uppercase flag negates the effect: "ffcccFsf" is equivalent to "cs".

The terminal program name for the "t" flag is taken from the environment variable $TERMCMD. If it doesn't exist, it tries to extract it from $TERM and uses "xterm" as a fallback if that fails.

Examples: `:open_with c` will open the file that you currently point at, even if you have selected other files. `:shell -w df` will run "df" and wait for you to press Enter before switching back to ranger.

PLUGINS

ranger's plugin system consists of python files which are located in *~/.config/ranger/plugins/* and are imported in alphabetical order when starting ranger. A plugin changes rangers behavior by overwriting or extending a function that ranger uses. This allows you to change pretty much every part of ranger, but there is no guarantee that things will continue to work in future versions as the source code evolves.

Adding new commands via a plugin as simple as specifying them like you would do in the *commands.py*.

There are some hooks that are specifically made for the use in plugins. They are functions that start with hook_ and can be found throughout the code.

```
grep 'def hook_' -r /path/to/rangers/source
```

Also try:

```
pydoc ranger.api
```

Note that you should NOT simply overwrite a function unless you know what you're doing. Instead, save the existing function and call it from your new one. This way, multiple plugins can use the same hook. There are several sample plugins in the */usr/share/doc/ranger/examples/* directory, including a hello-world plugin that describes this procedure.

KEY BINDINGS

Key bindings are defined in the file *ranger/config/rc.conf*. Check this file for a list of all key bindings. You can copy it to your local configuration directory with the --copy-config=rc option.

Many key bindings take an additional numeric argument. Type *5j* to move down 5 lines, *2l* to open a file in mode 2, *10<Space>* to mark 10 files.

This list contains the most useful bindings:

MAIN BINDINGS

h, j, k, l

Move left, down, up or right

^D or J, ^U or K

Move a half page down, up

H, L

Move back and forward in the history

gg

Move to the top

G

Move to the bottom

[,]

Move up and down in the parent directory.

^R

Reload everything

^L

Redraw the screen

i

Inspect the current file in a bigger window.

E

Edit the current file in $EDITOR ("nano" by default)

S

Open a shell in the current directory

?

Opens this man page

W

Opens the log window where you can review messages that pop up at the bottom.

w

Opens the task window where you can view and modify background processes that currently run in ranger. In there, you can type "dd" to abort a process and "J" or "K" to change the priority of a process. Only one process is run at a time.

^C

Stop the currently running background process that ranger has started, like copying files, loading directories or file previews.

<octal>=, +<who><what>, -<who><what>

Change the permissions of the selection. For example, `777=` is equivalent to `chmod 777 %s`, `+ar` does `chmod a+r %s`, `-ow` does `chmod o-w %s` etc.

yy

Copy (yank) the selection, like pressing Ctrl+C in modern GUI programs. (You can also type "ya" to add files to the copy buffer, "yr" to remove files again, or "yt" for toggling.)

dd

Cut the selection, like pressing Ctrl+X in modern GUI programs. (There are also "da", "dr" and "dt" shortcuts equivalent to "ya", "yr" and "yt".)

pp

Paste the files which were previously copied or cut, like pressing Ctrl+V in modern GUI programs.

po

Paste the copied/cut files, overwriting existing files.

pP, pO

Like pp and po, but queues the operation so that it will be executed *after* any other operations. Reminder: type `w` to open

the task window.

pl, pL

Create symlinks (absolute or relative) to the copied files

phl

Create hardlinks to the copied files

pht

Duplicate the subdirectory tree of the copied directory, then create hardlinks for each contained file into the new directory tree.

m*X*

Create a bookmark with the name *X*

`*X*

Move to the bookmark with the name *X*

n

Find the next file. By default, this gets you to the newest file in the directory, but if you search something using the keys /, cm, ct, ..., it will get you to the next found entry.

N

Find the previous file.

o*X*

Change the sort method (like in mutt)

z*X*

Change settings. See the settings section for a list of settings and their hotkey.

u*?*

Universal undo-key. Depending on the key that you press after "u", it either restores closed tabs (uq), removes tags (ut), clears the copy/cut buffer (ud), starts the reversed visual mode (uV) or clears the selection (uv).

f

Quickly navigate by entering a part of the filename.

Space

Mark a file.

v

Toggle the mark-status of all files

V

Starts the visual mode, which selects all files between the starting point and the cursor until you press ESC. To unselect files in the same way, use "uV".

/

Search for files in the current directory.

:

Open the console.

!

Open the console with the content "shell " so you can quickly run commands

@

Open the console with the content "shell %s", placing the cursor before the " %s" so you can quickly run commands with the

current selection as the argument.

r

Open the console with the content "open with " so you can decide which program to use to open the current file selection.

cd

Open the console with the content "cd "

Alt-*N*

Open a tab. N has to be a number from 0 to 9. If the tab doesn't exist yet, it will be created.

gn, ^N

Create a new tab.

gt, gT

Go to the next or previous tab. You can also use TAB and SHIFT+TAB instead.

gc, ^W

Close the current tab. The last tab cannot be closed this way.

M

A key chain that allows you to quickly change the line mode of all the files of the current directory. For a more permanent solution, use the command "default_linemode" in your rc.conf.

READLINE-LIKE BINDINGS IN THE CONSOLE

^B, ^F

Move left and right (B for back, F for forward)

^P, ^N

Move up and down (P for previous, N for Next)

^A, ^E

Move to the start or to the end

^D

Delete the current character.

^H

Backspace.

MOUSE BUTTONS

Left Mouse Button

Click on something and you'll move there. To run a file, "enter" it, like a directory, by clicking on the preview.

Right Mouse Button

Enter a directory or run a file.

Scroll Wheel

Scrolls up or down. You can point at the column of the parent directory while scrolling to switch directories.

SETTINGS

This section lists all built-in settings of ranger. The valid types for the value are in [brackets]. The hotkey to toggle the setting is in <brokets>, if a hotkey exists.

Settings can be changed in the file *~/.config/ranger/rc.conf* or on the fly with the command **:set option value**. Examples:

```
set column_ratios 1,2,3
set show_hidden true
```

Toggling options can be done with:

```
set show_hidden!
```

The different types of settings and an example for each type:

```
setting type   | example values
---------------+---------------------------
bool           | true, false
integer        | 1, 23, 1337
string         | foo, hello world
list           | 1,2,3,4
none           | none
```

You can view a list of all settings and their current values by pressing "3?" in ranger.

automatically_count_files [bool]

> Should ranger count and display the number of files in each directory as soon as it's visible? This gets slow with remote file sytems. Turning it off will still allow you to see the number of files after entering the directory.

autosave_bookmarks [bool]

> Save bookmarks (used with mX and `X) instantly? This helps to synchronize bookmarks between multiple ranger instances but leads to *slight* performance loss. When false, bookmarks are saved when ranger is exited.

autoupdate_cumulative_size [bool]

> You can display the "real" cumulative size of directories by using the command :get_cumulative_size or typing "dc". The size is expensive to calculate and will not be updated automatically. You can choose to update it automatically though by turning on this option.

cd_bookmarks [bool]

Specify whether bookmarks should be included in the tab completion of the "cd" command.

collapse_preview [bool] <zc>

When no preview is visible, should the last column be squeezed to make use of the whitespace?

colorscheme [string]

Which colorscheme to use? These colorschemes are available by default: **default**, **jungle**, **snow**. Snow is a monochrome scheme, jungle replaces blue directories with green ones for better visibility on certain terminals.

column_ratios [list]

How many columns are there, and what are their relative widths? For example, a value of 1,1,1 would mean 3 evenly sized columns. 1,1,1,1,4 means 5 columns with the preview column being as large as the other columns combined.

confirm_on_delete [string]

Ask for a confirmation when running the "delete" command? Valid values are "always" (default), "never", "multiple". With "multiple", ranger will ask only if you delete multiple files at once.

dirname_in_tabs [bool]

Display the directory name in tabs?

display_size_in_main_column [bool]

Display the file size in the main column?

display_size_in_status_bar [bool]

Display the file size in the status bar?

display_tags_in_all_columns [bool]

Display tags in all columns?

draw_borders [bool]

Draw borders around columns?

draw_progress_bar_in_status_bar [bool]

Draw a progress bar in the status bar which displays the average state of all currently running tasks which support progress bars?

flushinput [bool] <zi>

Flush the input after each key hit? One advantage is that when scrolling down with "j", ranger stops scrolling instantly when you release the key. One disadvantage is that when you type commands blindly, some keys might get lost.

hidden_filter [string]

A regular expression pattern for files which should be hidden. For example, this pattern will hide all files that start with a dot or end with a tilde.

```
set hidden_filter ^\.|~$
```

idle_delay [integer]

The delay that ranger idly waits for user input, in milliseconds, with a resolution of 100ms. Lower delay reduces lag between directory updates but increases CPU load.

line_numbers [string]

Show line numbers in main column. Possible values are:

```
false      turn the feature off
absolute   absolute line numbers for use with "<N>gg"
relative   relative line numbers for "<N>k" or "<N>j"
```

max_console_history_size [integer, none]

How many console commands should be kept in history? "none" will disable the limit.

max_history_size [integer, none]

How many directory changes should be kept in history?

metadata_deep_search [bool]

When the metadata manager module looks for metadata, should it only look for a ".metadata.json" file in the current directory, or do a deep search and check all directories above the current one as well?

mouse_enabled [bool] <zm>

Enable mouse input?

padding_right [bool]

When collapse_preview is on and there is no preview, should there remain a little padding on the right? This allows you to click into that space to run the file.

preview_directories [bool] <zP>

Preview directories in the preview column?

preview_files [bool] <zp>

Preview files in the preview column?

preview_images [bool]

Draw images inside the console with the external program w3mimgpreview?

preview_max_size [int]

Avoid previewing files that exceed a certain size, in bytes. Use a

value of 0 to disable this feature.

preview_script [string, none]

Which script should handle generating previews? If the file doesn't exist, or use_preview_script is off, ranger will handle previews itself by just printing the content.

save_console_history [bool]

Should the console history be saved on exit? If disabled, the console history is reset when you restart ranger.

scroll_offset [integer]

Try to keep this much space between the top/bottom border when scrolling.

shorten_title [integer]

Trim the title of the window if it gets long? The number defines how many directories are displayed at once. A value of 0 turns off this feature.

show_cursor [bool]

Always show the terminal cursor?

show_hidden_bookmarks [bool]

Show dotfiles in the bookmark preview window? (Type ')

show_hidden [bool] <zh>, <^H>

Show hidden files?

sort_case_insensitive [bool] <zc>

Sort case-insensitively? If true, "a" will be listed before "B" even though its ASCII value is higher.

sort_directories_first [bool] <zd>

Sort directories first?

sort_reverse [bool] <or>

Reverse the order of files?

sort_unicode [bool]

When sorting according to some string, should the unicode characters be compared, instead of looking at the raw character values to save time?

sort [string] <oa>, <ob>, <oc>, <oe>, <om>, <on>, <ot>, <os>, <oz>

Which sorting mechanism should be used? Choose one of **atime**, **basename**, **ctime**, **extension**, **mtime**, **natural**, **type**, **size**, **random**

Note: You can reverse the order by typing an uppercase second letter in the key combination, e.g. "oN" to sort from Z to A.

status_bar_on_top [bool]

Put the status bar at the top of the window?

tilde_in_titlebar [bool]

Abbreviate $HOME with ~ in the title bar (first line) of ranger?

unicode_ellipsis [bool]

Use a unicode "..." character instead of "~" to mark cut-off filenames?

update_title [bool]

Set a window title?

update_tmux_title [bool]

Set the title to "ranger" in the tmux program?

use_preview_script [bool] <zv>

Use the preview script defined in the setting *preview_script*?

vcs_aware [bool]

Gather and display data about version control systems. Supported vcs: git, hg.

vcs_backend_git, vcs_backend_hg, vcs_backend_bzr [string]

Sets the state for the version control backend. The possible values are:

```
disabled   don't display any information.
local      display only local state.
enabled    display both, local and remote state. May be slow for hg and bzr.
```

xterm_alt_key [bool]

Enable this if key combinations with the Alt Key don't work for you. (Especially on xterm)

clear_filters_on_dir_change [bool]

If set to 'true', persistent filters would be cleared upon leaving the directory

COMMANDS

You can enter the commands in the console which is opened by pressing ":".

You can always get a list of the currently existing commands by typing "2?" in ranger. For your convenience, this is a list of the "public" commands including their parameters, excluding descriptions:

```
alias [newcommand] [oldcommand]
bulkrename
cd [directory]
chain command1[; command2[; command3...]]
chmod octal_number
cmap key command
```

```
console [-pSTARTPOSITION] command
copycmap key newkey [newkey2...]
copymap key newkey [newkey2...]
copypmap key newkey [newkey2...]
copytmap key newkey [newkey2...]
cunmap keys...
default_linemode [path=regexp | tag=tags] linemodename
delete
echo [text]
edit [filename]
eval [-q] python_code
filter [string]
filter_inode_type [dfl]
find pattern
flat level
grep pattern
help
linemode linemodename
load_copy_buffer
map key command
mark pattern
mark_tag [tags]
meta key value
mkdir dirname
open_with [application] [flags] [mode]
pmap key command
prompt_metadata [key1 [key2 [...]]]
punmap keys...
quit
quit!
relink newpath
rename_append
rename newname
save_copy_buffer
scout [-FLAGS] pattern
search pattern
search_inc pattern
set option value
setintag tags option value
setlocal [path=<path>] option value
shell [-FLAGS] command
source filename
terminal
tmap key command
touch filename
travel pattern
tunmap keys...
unmap keys...
unmark pattern
unmark_tag [tags]
```

There are additional commands which are directly translated to python functions, one for every method in the ranger.core.actions.Actions class. They are not documented here, since they are mostly for key bindings, not to be typed in by a user. Read the source if you are interested in them.

These are the public commands including their descriptions:

alias [*newcommand*] [*oldcommand*]

Copies the oldcommand as newcommand.

bulkrename

This command opens a list of selected files in an external editor. After you edit and save the file, it will generate a shell script which does bulk renaming according to the changes you did in the file.

This shell script is opened in an editor for you to review. After you close it, it will be executed.

cd [*directory*]

The cd command changes the directory. The command `:cd -` is equivalent to typing ``.

chain *command1*[; *command2*[; *command3*...]]

Combines multiple commands into one, separated by semicolons.

chmod *octal_number*

Sets the permissions of the selection to the octal number.

The octal number is between 000 and 777. The digits specify the permissions for the user, the group and others. A 1 permits execution, a 2 permits writing, a 4 permits reading. Add those numbers to combine them. So a 7 permits everything.

Key bindings in the form of [-+]<who><what> and <octal>= also exist. For example, **+ar** allows reading for everyone, -ow forbids others to write and 777= allows everything.

See also: man 1 chmod

cmap *key command*

Binds keys for the console. Works like the `map` command.

console [-p*N*] *command*

Opens the console with the command already typed in. The cursor is placed at *N*.

copycmap *key newkey* [*newkey2* ...]

See `copymap`

copymap *key newkey* [*newkey2* ...]

Copies the keybinding *key* to *newkey* in the "browser" context. This is a deep copy, so if you change the new binding (or parts of it) later, the old one is not modified.

To copy key bindings of the console, taskview, or pager use "copycmap", "copytmap" or "copypmap".

copypmap *key newkey* [*newkey2* ...]

See `copymap`

copytmap *key newkey* [*newkey2* ...]

See `copymap`

cunmap [*keys...*]

Removes key mappings of the console. Works like the `unmap` command.

default_linemode [*path=regexp* | *tag=tags*] *linemodename*

Sets the default linemode. See *linemode* command.

Examples:

Set the global default linemode to "permissions":
:default_linemode permissions

Set the default linemode to "permissions" for all files tagged with "p" or "P": :default_linemode tag=pP permissions

Set the default linemode for all files in ~/books/ to "metatitle": :default_linemode path=/home/.*?/books/.* metatitle

delete

Destroy all files in the selection with a roundhouse kick. ranger will ask for a confirmation if you attempt to delete multiple (marked) files or non-empty directories. This can be changed by modifying the setting "confirm_on_delete".

echo *text*

Display the text in the statusbar.

edit [*filename*]

Edit the current file or the file in the argument.

eval [*-q*] *python_code*

Evaluates the python code. `fm' is a reference to the FM instance. To display text, use the function `p'. The result is displayed on the screen unless you use the "-q" option.

Examples: :eval fm :eval len(fm.tabs) :eval p("Hello World!")

filter [*string*]

Displays only the files which contain the *string* in their basename. Running this command without any parameter will reset the filter.

This command is based on the *scout* command and supports all of its options.

filter_inode_type [dfl]

Displays only the files of specified inode type. To display only directories, use the 'd' parameter. To display only files, use the 'f' parameter. To display only links, use the 'l' parameter. Parameters can be combined. To remove this filter, use no

parameter.

find *pattern*

Search files in the current directory that contain the given (case-insensitive) string in their name as you type. Once there is an unambiguous result, it will be run immediately. (Or entered, if it's a directory.)

This command is based on the *scout* command and supports all of its options.

flat level

Flattens the directory view up to the specified level. Level -1 means infinite level. Level 0 means standard view without flattened directory view. Level values -2 and less are invalid.

grep *pattern*

Looks for a string in all marked files or directories.

help

Provides a quick way to view ranger documentations.

linemode *linemodename*

Sets the linemode of all files in the current directory. The linemode may be:

```
"filename": display each line as "<basename>...<size>"
"fileinfo": display each line as "<basename>...<file(1) output>"
"permissions": display each line as "<permissions> <owner> <group> <basename>"
"metatitle": display metadata from .metadata.json files if
    available, fall back to the "filename" linemode if no
    metadata was found.  See :meta command.
```

The custom linemodes may be added by subclassing the *LinemodeBase* class. See the *ranger.core.linemode* module for some examples.

load_copy_buffer

Load the copy buffer from *~/.config/ranger/copy_buffer*. This can be used to pass the list of copied files to another ranger instance.

map *key command*

Assign the key combination to the given command. Whenever you type the key/keys, the command will be executed. Additionally, if you use a quantifier when typing the key, like 5j, it will be passed to the command as the attribute "self.quantifier".

The keys you bind with this command are accessible in the file browser only, not in the console, task view or pager. To bind keys there, use the commands "cmap", "tmap" or "pmap".

mark *pattern*

Mark all files matching the regular expression pattern.

This command is based on the *scout* command and supports all of its options.

mark_tag [*tags*]

Mark all tags that are tagged with either of the given tags. When leaving out the tag argument, all tagged files are marked.

meta *key value*

Set the metadata of the currently highlighted file. Example:

```
:meta title The Hitchhiker's Guide to the Galaxy
:meta year 1979
```

This metadata can be displayed by, for example, using the "metatitle" line mode by typing Mt.

mkdir *dirname*

Creates a directory with the name *dirname*.

open_with [*application*] [*flags*] [*mode*]

Open the selected files with the given application, unless it is omitted, in which case the default application is used. *flags* change the way the application is executed and are described in their own section in this man page. The *mode* is a number that specifies which application to use. The list of applications is generated by the external file opener "rifle" and can be displayed when pressing "r" in ranger.

Note that if you specify an application, the mode is ignored.

pmap *key command*

Binds keys for the pager. Works like the `map` command.

prompt_metadata [*keys ...*]

Prompt the user to input metadata with the `meta` command for multiple keys in a row.

punmap [*keys ...*]

Removes key mappings of the pager. Works like the `unmap` command.

quit

Like quit!, but closes only this tab if multiple tabs are open.

quit!

Quit ranger. The current directory will be bookmarked as ' so you can re-enter it by typing `` or '' the next time you start ranger.

relink *newpath*

Change the link destination of the current symlink file to <newpath>. First <tab> will load the original link.

rename_append

Opens the console with ":rename <current file>" with the cursor automatically placed before the file extension

rename *newname*

Rename the current file. If a file with that name already exists, the renaming will fail. Also try the key binding A for appending something to a file name.

save_copy_buffer

Save the copy buffer to *~/.config/ranger/copy_buffer*. This can be used to pass the list of copied files to another ranger instance.

scout [-*flags*...] [--] *pattern*

Swiss army knife command for searching, traveling and filtering files. The command takes various flags as arguments which can be used to influence its behaviour:

```
-a = automatically open a file on unambiguous match
-e = open the selected file when pressing enter
-f = filter files that match the current search pattern
-g = interpret pattern as a glob pattern
-i = ignore the letter case of the files
-k = keep the console open when changing a directory with the command
-l = letter skipping; e.g. allow "rdme" to match the file "readme"
-m = mark the matching files after pressing enter
-M = unmark the matching files after pressing enter
-p = permanent filter: hide non-matching files after pressing enter
-r = interpret pattern as a regular expression pattern
-s = smart case; like -i unless pattern contains upper case letters
-t = apply filter and search pattern as you type
-v = inverts the match
```

Multiple flags can be combined. For example, ":scout -gpt" would create a :filter-like command using globbing.

search *pattern*

Search files in the current directory that match the given (case insensitive) regular expression pattern.

This command is based on the *scout* command and supports all of its options.

search_inc *pattern*

Search files in the current directory that match the given (case

insensitive) regular expression pattern. This command gets you to matching files as you type.

This command is based on the *scout* command and supports all of its options.

set *option value*

Assigns a new value to an option. Valid options are listed in the settings section. Use tab completion to get the current value of an option, though this doesn't work for functions and regular expressions. Valid values are:

```
setting type   | example values
---------------+---------------------------
bool           | true, false
integer        | 1, 23, 1337
string         | foo, hello world
list           | 1,2,3,4
none           | none
```

setintag *tags option value*

Assigns a new value to an option, but locally for the directories that are marked with *tag*. This means, that this option only takes effect when visiting that directory.

For example, to change the sorting order in your downloads directory, tag it with the *v* tag by typing *"v*, then use this command:

```
setintag v sort ctime
```

setlocal [path=*path*] *option value*

Assigns a new value to an option, but locally for the directory given by *path*. This means, that this option only takes effect when visiting that directory. If no path is given, uses the current directory.

path is a regular expression. This means that `path=~/dl` applies to all paths that start with *~/dl*, e.g. *~/dl2* and *~/dl/foo*. To avoid this, use `path=~/dl$`.

shell [-*flags*] *command*

Run a shell command. *flags* are discussed in their own section.

source *filename*

Reads commands from a file and executes them in the ranger console.

This can be used to re-evaluate the rc.conf file after changing it:

```
map X chain shell vim -p %confdir/rc.conf %rangerdir/config/rc.conf; source %confdir/rc.conf
```

terminal

Spawns the *x-terminal-emulator* starting in the current directory.

tmap *key command*

Binds keys for the taskview. Works like the `map` command.

touch *filename*

Creates an empty file with the name *filename*, unless it already exists.

travel *pattern*

Filters the current directory for files containing the letters in the string, possibly with other letters in between. The filter is applied as you type. When only one directory is left, it is entered and the console is automatically reopened, allowing for fast travel. To close the console, press ESC or execute a file.

This command is based on the *scout* command and supports all of its options.

tunmap [*keys ...*]

Removes key mappings of the taskview. Works like the `unmap` command.

unmap [*keys* ...]

Removes the given key mappings in the "browser" context. To

unmap key bindings in the console, taskview, or pager use "cunmap", "tunmap" or "punmap".

unmark *pattern*

Unmark all files matching a regular expression pattern.

This command is based on the *scout* command and supports all of its options.

unmark_tag [*tags*]

Unmark all tags that are tagged with either of the given tags. When leaving out the tag argument, all tagged files are unmarked.

FILES

ranger reads several configuration files which are located in *$HOME/.config/ranger* or *$XDG_CONFIG_HOME/ranger* if $XDG_CONFIG_HOME is defined. You can use the --copy-config option to obtain the default configuration files. Each of the files contains further documentation.

You don't need to copy the whole file though, most configuration files are overlaid on top of the defaults (*commands.py*, *rc.conf*) or can be sub-classed (*colorschemes*).

That being said, the user configuration files *rc.conf* and *commands.py* are loaded only after ranger loads the default configuration files. This may lead to some confusing situations, for example when a key is being bound despite the corresponding line being removed from the user's copy of the configuration file. This behavior may be disabled with an environment variable (see also: **ENVIRONMENT**).

When starting ranger with the **--clean** option, it will not access or create any of these files.

CONFIGURATION

rc.conf

Contains a list of commands which are executed on startup. Mostly key bindings and settings are defined here.

commands.py

A python module that defines commands which can be used in ranger's console by typing ":" or in the rc.conf file. Note that you can define commands in the same manner within plugins.

commands_full.py

This file is copied by --copy-config=commands_full and serves as a reference for custom commands. It is entirely ignored by ranger.

rifle.conf

This is the configuration file for the built-in file launcher called "rifle".

scope.sh

This is a script that handles file previews. When the options *use_preview_script* and *preview_files* are set, the program specified in the option *preview_script* is run and its output and/or exit code determines rangers reaction.

colorschemes/

Colorschemes can be placed here.

plugins/

Plugins can be placed here.

STORAGE

bookmarks

This file contains a list of bookmarks. The syntax is /^(.):(.*)$/. The first character is the bookmark key and the rest after the colon is the path to the file. In ranger, bookmarks can be set by typing m<key>, accessed by typing '<key> and deleted by typing um<key>.

copy_buffer

When running the command :save_copy_buffer, the paths of all currently copied files are saved in this file. You can later run :load_copy_buffer to copy the same files again, pass them to another ranger instance or process them in a script.

history

Contains a list of commands that have been previously typed in.

tagged

Contains a list of tagged files. The syntax is /^(.:)?(.*)$/ where the first letter is the optional name of the tag and the rest after the optional colon is the path to the file. In ranger, tags can be set by pressing t and removed with T. To assign a named tag, type "<tagname>.

ENVIRONMENT

These environment variables have an effect on ranger:

RANGER_LEVEL

ranger sets this environment variable to "1" or increments it if it already exists. External programs can determine whether they were spawned from ranger by checking for this variable.

RANGER_LOAD_DEFAULT_RC

If this variable is set to FALSE, ranger will not load the default

rc.conf. This can save time if you copied the whole rc.conf to ~/.config/ranger/ and don't need the default one at all.

EDITOR

Defines the editor to be used for the "E" key. Defaults to "nano".

SHELL

Defines the shell that ranger is going to use with the :shell command and the "S" key. Defaults to "/bin/sh".

TERMCMD

Defines the terminal emulator command that ranger is going to use with the :terminal command and the "t" run flag. Defaults to "xterm".

XDG_CONFIG_HOME

Specifies the directory for configuration files. Defaults to *$HOME/.config*.

PYTHONOPTIMIZE

This variable determines the optimize level of python.

Using PYTHONOPTIMIZE=1 (like python -O) will make python discard assertion statements. You will gain efficiency at the cost of losing some debug info.

Using PYTHONOPTIMIZE=2 (like python -OO) will additionally discard any docstrings. Using this will disable the <F1> key on commands.

W3MIMGDISPLAY_PATH

By changing this variable, you can change the path of the executable file for image previews. By default, it is set to */usr/lib/w3m/w3mimgdisplay*.

EXAMPLES

There are various examples on how to extend ranger with plugins or combine ranger with other programs. These can be found in the */usr/share/doc/ranger/examples/* directory, or the *doc/ranger/* that is provided along with the source code.

LICENSE

GNU General Public License 3 or (at your option) any later version.

LINKS

Download: http://ranger.nongnu.org/ranger-stable.tar.gz
The project page: http://ranger.nongnu.org/
The mailing list: http://savannah.nongnu.org/mail/?group=ranger
IRC channel: #ranger on freenode.net

ranger is maintained with the git version control system. To fetch a fresh copy, run:

```
git clone git://git.savannah.nongnu.org/ranger.git
```

SEE ALSO

rifle(1)

BUGS

Report bugs here: https://github.com/hut/ranger/issues

Please include as much relevant information as possible. For the most diagnostic output, run ranger like this: `PYTHONOPTIMIZE= ranger --debug`