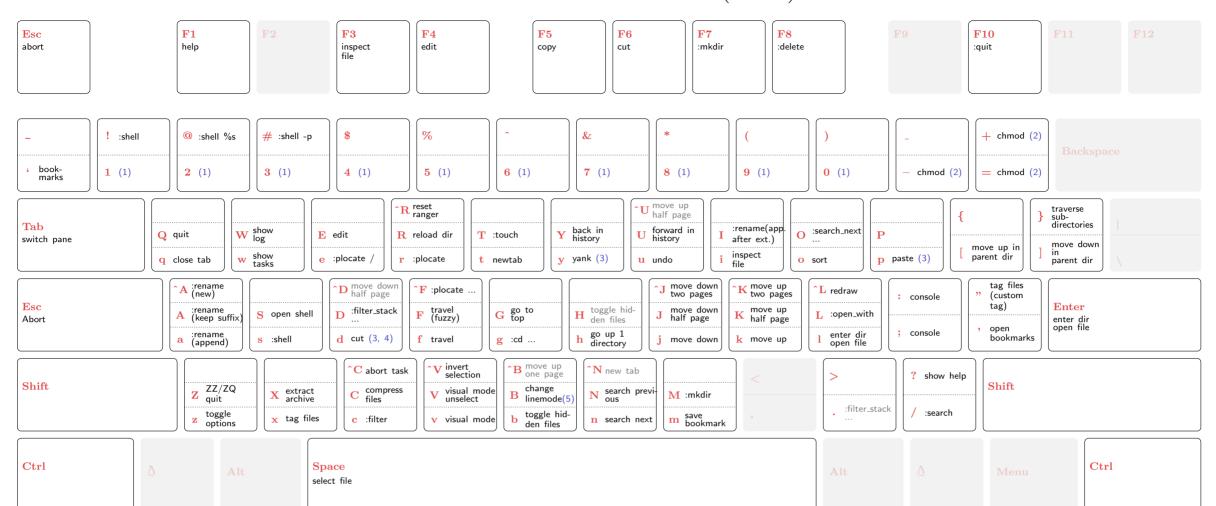
RANGER KEYBINDINGS (HAUG)



Macros

Macros can be used in commands. They are like global variables with dynamic content.

- %f The base name of the current file
- %d The path of the current directory
- $\bullet~\%s$ The names of the currently selected files
- $\bullet~\%t$ The names of all tagged files in this directory
- %c The paths of the currently copied files
- %any The key used in a key binding with <any>. Example: map x<any> shell -w echo %any
- %rangerdir The path to the ranger python module
- %space Just a space, to avoid typing trailing spaces

Example: map yp shell echo %d/%f | xsel -i They can be escaped by replacing % with %%.

Hints

Various helpful things

• :eval fm.copy_buffer.clear() Empties the copy buffer (maybe there is a better method though)

Config files

run ranger --copy-config=all to copy the default config files to ~/.config/ranger/.

- rc.conf A list of commands that are executed when ranger starts. Options, key bindings and aliases are found here. Pro tip: Adding export RANGER_LOAD_DEFAULT_RC=FALSE to your shell rc will skip loading the default rc.conf before your own.
- commands.py A python script containing custom commands
- rifle.conf Rules for rifle, the file opener. Each lines looks like list of conditions = command. When ranger opens a file, it tests those conditions. The first command where all conditions are true will be exe-
- scope.sh The script that generates file previews. Plugins can be put in the plugins/ subdirectory, colorschemes in colorschemes/. See /usr/share/doc/ranger/examples.

Commands

Commands can be typed in by pressing: or added to _/.config/ranger/rc.conf to apply then whenever ranger starts. All commands are listed in the man page. Some important ones:

- :shell [<flags>] <command> calls the given <command> with the shell specified in the environment variable \$SHELL. <flags> can be f to fork the process or p to pipe the output to a pager. Macros like %f and %s are especially useful here. Example: :shell -f inkscape %f or :shell sudo cp
- :alias <new> <old> creates the command <new> that calls <old>. The neat thing is that you can pass arguments to the next command. Example: :alias touch shell touch will allow you to type :touch FILE, which will be translated to :shell touch FILE.
- :map <key> <command> makes the <key> run <command> when pressed. This is the typical way todefine key bindings in rc.conf. There is also pmap to define keys in the pager and unmap and punmap to remove key bindings.

(1) quantifiers

numbers can be used as a quantifier in various commands, for example 5j will movethe cursor down 5 by lines, 3<space> selects 3 files, 4<TAB> moves you to the 4th tab.

(2) chmod

the keys -, - and = change the permissions of files. See man chmod.

- [+-][augo][rwxXst] (e.g. +gw means "add write permissions to the group)
- [+-] [rwxXst] (e.g. -x means "remove execute permissions from everybody")
- <octal>= (e.g. 777= means "give full permissions to everybody")

(3) yank, copy, paste

To copy files, select them with the cursor (or <space>, in case of multiple files). Type dd (to cut) or yy (to copy). Move to the destination and type pp. Type da (or ya) to add files to the copy buffer, allowing you to copy from multiple folders.

(4) d*

d also starts the keybindings

- dc (calculate size of the content of a directory)
- du/dU (calculate directory size with the du program)
- dD (open the console with ":delete")

(5) linemode

 ${\tt M{<}key{>}}$ changes the line mode, the way files are drawn.

- Mf draws just the file name
- ullet Mp draws permissions
- Mi draws file type information
- Mt draws metadata, as defined with the :meta command.

You can add custom linemodes as described in /usr/share/doc/ranger/examples/plugin_linemode.py.