

Linux Administration

Editing files

Xavier Belanger

**This work is licensed under
a Creative Commons Attribution-ShareAlike 4.0 International License.**

<http://creativecommons.org/licenses/by-sa/4.0/>

You are free to:

- **Share** — copy and redistribute the material in any medium or format
- **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Interactive and programmatic editing

Two ways are available to edit text files:

- using a text editor in interactive mode
- using various commands to make changes based on patterns

Text editors

- Plenty of text editors are available on Linux, both in text mode or graphical mode.
- In text mode, the most popular choices are vi/vim, emacs and nano.
- nano is probably the best option to start with for beginners.

Using nano

- You can launch nano to create a brand new file, or specify the file to create or to open.
 - *nano*
 - *nano <file>*
- The most useful commands are listed in the status bar:
 - ^O = *ctrl* + *o* to save
 - ^X = *ctrl* + *x* to exit
 - *M-U* = *alt* + *u* to cancel an action (undo)
 - ^K = *ctrl* + *k* to cut a line
 - ^U = *ctrl* + *u* to paste a line

Using vi

- *vi* (or its more modern clone *vim*) is a more powerful text editor, with different modes.
- You can launch *vi* to create a brand new file, or specify the file to create or to open.
 - *vi*
 - *vi <file>*
- The two main modes are “normal” to execute commands, and “insert” to edit text directly.
- You can switch to the insert mode with the “i” (insert) or “a” (append) commands; you can go back to the normal mode by using the escape (esc) key.

vim commands

The following commands are to be used in normal mode.

- `:w` write to a file
- `:wq` write to a file and quit
- `:u` undo an action
- `yy` copy a line
- `p` paste a line
- `dd` delete a line
- `/<pattern>` search for a pattern
- `n / N` next / previous pattern match

EDITOR and VISUAL variables

- Various commands can be used to edit specific files (*vipw*, *crontab -e*, ...) and will start with a default text editor.
- You can set the EDITOR and VISUAL variables to define your preferences.
 - *EDITOR=vim*
 - *VISUAL=vim*
 - *export EDITOR VISUAL*
- Debian-based Linux distributions also provide the *update-alternatives* command to define a default text editor.

The `tr` command

- `tr` stands for “translate” and will convert a set of characters to another.
- The following command will convert all lowercase characters to uppercase from one file and save it to another:
- `cat <file 1> | tr 'a-z' 'A-Z' > <file 2>`

The sort command

This command will perform some type of sorting on the target file:

- Normal sorting: `sort <file>`
- Reverse sorting: `sort -r <file>`
- Numerical sorting: `sort -n <file>`
- Version sorting: `sort -V <file>`

The `uniq` command

- *uniq* will delete duplicated lines in a file:
- *uniq* <file>
- One useful option is to count (and not delete) occurrences of each line:
- *uniq -c* <file>

The head and tail commands

- *head* will display the ten first lines of a file, *tail* the last ten.
- You can specify the number of lines to display with the `-n` option.
- *tail -f <file>* can be used to see how new content gets added in real time.

The cut command

- *cut* will split the content of a file based a separator.
- The following example extract only the username from the passwd file:
- *cut -d ":" -f 1 /etc/passwd*

The `paste` command

- *paste* will combine multiple files into one output, values are separated by tabs by default.
- *paste -d “.” <file1> <file2>*

The sed command

- *sed* is a stream editor, it can be used to process files line by line.
- Substitute content:
 - *sed 's/<pattern>/<replacement/g' <file>*
- Deleting 10 first lines:
 - *sed '1,10d' <file>*
- The *-i* option will make changes in-place (into the same file).