Overview of Linux commands

Starting out on the shell

This document will describe a brief overview of some common Linux commands that you may use in a `bash` shell. They should be available on most, if not all, Linux systems. We will use the following syntax:

```
    command <mandatory argument> [optional argument]
```

Note that you won't include the `<>` or `[]` in the actual command.

Many commands also have options that can switch certain behaviour. These are usually specified as letters or words prefixed by dashes. For example, `ls -l` lists the attributes of the files in the current directory. Similarly, `ls -a` (short for `ls --all`) includes hidden files. It is possible to combine such options: `ls -l -a` or even `ls -la`.

File system

Command	Explanation
`pwd`	**P**rint current **W**orking **D**irectory
`ls [dir]`	List contents of current working directory or `dir`
`ls -l [dir]`	Also show file attributes, like owner, size or mode
`ls -a [dir]`	Also show hidden files
`mkdir <dir>`	Create new directory `dir`
`cd [dir]`	**C**hange current working **D**irectory to `dir`, or to the *home directory* if not specified
`cp <src> <target>`	Copy file `source` to `target`. `target` may be a directory or a filename.
`cp <s1> <s2...> <dir>`	Copy specified source files to directory `dir`.
`cp -r <dir> <target>`	Copy specified directory `dir` *recursively* to `target`.
`mv <src> <target>`	Move file into `target` or rename it `target`.
`rm <file>`	Remove `file`
`rmdir <dir>`	Remove empty directory `dir`
`rm -r <dir>`	Remove directory `dir` and all that is in it.
`touch <file>`	Create `file` if it does not exist or update the last-modified time.

Being lazy

Item	Explanation
the tab key	Autocomplete command, file or directory name
tab twice	See available commands
`*`	Any string, for example `cp a* backup`
`?`	Any character, for example `ls foo?ar`
`history`	Show previous commands
arrow keys up/down	Scroll through previous commands
`!!`	Repeat last command (`echo !!`)
`!pattern`	Repeat last command starting with *pattern*. (`!cp`)

If you want to copy paste, use the context menu. Often the shortcut is `Ctrl+Shift+`{`C,V`}; note that `Ctrl+C` **terminates** the current process!

Abbreviations for paths

Item	Explanation
`~`	Home directory
`.`	Current working directory, e.g. `cp /tmp/test.txt .`
`..`	Parent directory, e.g. `cd ..`
`/`	Root-directory of the filesystem, e.g. `ls /`.

Process control

Item	Explanation
`ps`	Show a list of processes
`Ctrl+C`	Terminate current process
`kill <pid>`	Tell process with id `pid` to stop
`kill -KILL <pid>`	Instruct the operating system to commit murder on process with id `pid`
`killall <process name>`	Kill all processes with that name

Input and output

Item	Explanation
`<command> > <file>`	Send output from `command` to `file`
`<command> < <file>`	Send `file` as input to `file`
`<command1> | <command2>`	Send output from `command1` to `command2`

Session control

Item	Explanation
`ssh <hostname>`	Start a remote shell on the specified host, e.g. `ssh lilo.science.ru.nl`. `ssh` will try to log in with the same username as on your current machine.
`ssh <user>@<hostname>`	Log in as `user` on the specified host
`logout`	Log out from the current login shell
`exit`	Exit current shell

Learning more

Item	Explanation
`man <command>`	Show manual for `command`
`<command> --help`	Most commands will show you a brief overview of their usage
`<command> -h`	`-h` is usually short for `--help`
`which <command>`	Will tell you the location of `command` on the file system

Convenient tools

Item	Explanation
`cat <file>`	Show contents of `file`. If you specify multiple files it will concatenate them.
`grep <pattern> <file>`	Show all lines that contain `pattern` in `file`.
`grep <pattern>`	Show all lines that contain `pattern` from *standard input*, e.g. `cat file | grep pattern`
`grep -v <pattern>`	Inverse, show all line that *do not* match `pattern`.
`grep -i <pattern>`	Search for `pattern` case-insensitively.
`grep -o <pattern>`	Show **o**nly the matching bits of the input.
`grep -r <pattern> <dir>`	Recursively find all occurrences of `pattern` in `dir`
`tr 'A' 'B'`	Replace all occurrences of `'A'` by `'B'`. Reads standard input, so use for example as `cat file | tr A B`
`tr -d<char>`	Deletes all occurrences of `char`. Reads standard input, so use for example as `cat file | tr -d A`
`less [file]`	Show `file` or *standard input* page-by-page.
`file <file>`	Show what kind of file `file` is.
`wc <file>`	Count the number of lines, words and characters in `file`. (**w**ord **c**ount)
`head <file>`	Show the first 10 lines from `file`. Use `-n<num>` option for other amounts.
`tail <file>`	Show the last 10 lines from `file`. Use `-n<num>` option for other amounts.
`tail -f <file>`	Show the last 10 lines from `file` and then show new lines as they get added to `file`. Useful for logs.
`diff <file1> <file2>`	Show the differences between two files.

Editors

You can find “Text Editor” in the menu of the Ubuntu desktops, but you may also try the following (in order of how complicated the editor is).

Item	Explanation
`gedit [file]`	Start the GNOME text editor, opening `file`
`nano [file]`	Start the `nano` editor, opening `file`
`pico [file]`	Start the `pico` editor, opening `file`
`vim [file]`	Start the `vim` editor, opening `file`
`gvim [file]`	Start the graphical version of `vim`, opening `file`

You may hear about `vim` a lot, as it is a popular, very powerful editor. It does have a steep learning curve, however. You may enjoy this game that explains how it works: `https://vim-adventures.com/`.

Archiving

Item	Explanation
`tar czvf <file.tar.gz> <dir>`	Create a tar archive from the files in `dir`, and compress these with the gzip algorithm into `file.tar.gz`. `czvf` is a contraction of the flags **c**reate, **z**ip, **v**erbose (lots of output) and into **f**ile `file.tar.gz`.
`tar xzvf <file.tar.gz>`	Extract the archive `file.tar.gz`. `xzvf` is a contraction of e**x**tract, g**z**ip, **v**erbose and read from **f**ile.

Compile

Item	Explanation
`gcc <file> -o <target>`	Compile C program `file` and put the output on `target`
`make`	Run the `Makefile` in the current directory.