1 Basic linux commands

Here some basic linux commands will be briefly explained. For information about more commands, feel free to consult with the internet.

1.1 Getting around

Command	Explanation
`ls`	list contents of current directory
`ls` *dirname*	list contents of directory named *dirname*
`cd` *dirname*	change current directory to *dirname*
`cd ~`	go to home directory (default directory)
`cd ..`	go one directory up
`pwd`	print name of current directory
`mkdir` *dirname*	make a new directory named *dirname*
`man` *command*	show manual for *command*

1.2 File manipulation

Command	Explanation
`cp` *filename1 filename2*	copy file
`cp -r` *dirname1 dirname2*	copy directory
`mv` *name1 name2*	move file or directory (can be used for renaming)
`rm` *filename*	remove file (you won't be able to recover removed file)
`rm -r` *dirname*	remove directory and its contents
`more` *filename*	command for paging through text file one screenful at a time
`less` *filename*	similar as `more`. Better for viewing large text files (`q` to quit)
`cat` *filename1 filename2 filenameN*	concatenate text files in print output to terminal window
`head` *filename*	print first 10 lines of a text file to terminal window
`head -n 20` *filename*	print first 20 lines of a text file to terminal window
`tail`	opposite of `head`
`wc` *filename*	show the number of lines, words and characters in a text file
`cut -f 2` *tabDelimitedFilename*	extract 2nd column from a tab delimited file
`cut -f 3 -d ,` *comaSeperatedFilename*	extract 3rd column from a coma seperated file
`nano` *filename*	open *filename* in text editor `nano`

1.3 Archiving and unarchiving of files

Note that for the `tar` utility, option `c` stands for compress, `x` - for uncompress or extract, `z` - for dealing with *tar.gz*, and `j` - for dealing with *tar.bz2*

1.3.1 Compressing

Command	Explanation
`tar -cvf` *filename.tar filename*	**compress** file to **.tar** format
`tar -zcvf` *filename.tar.gz filename*	**compress** file to **.tar.gz** format
`tar -jcvf` *filename.tar.bz2 filename*	**compress** file to **.tar.bz2** format
`zip` *filename.zip filename*	**compress** file to **.zip** format
`gzip` *filename*	**compress** file to **.gz** format

1.3.2 Uncompressing

Command	Explanation
`tar -xvf` *filename.tar*	**uncompress** from **.tar** format
`tar -zxvf` *filename.tar.gz*	**uncompress** from **.tar.gz** format
`tar -jxvf` *filename.tar.bz2*	**uncompress** from **.tar.bz2** format
`unzip` *filename.zip*	**uncompress** from **.zip** format
`gzip -d` *file.gz*	**uncompress** from **.gz** format

1.4 Input/Output redirection

Command	Explanation
command `>` *filename*	output of *command* is saved to *filename*, overwriting it
command `>>` *filename*	output of *command* is appended at the end of *filename*
command `<` *filename*	*command* reads input from *filename*
command1 `|` *command2*	*command2* takes the output of *command1* and produces result

1.5 Lists of commands

Command	Explanation
command1 `;` *command2*	*command2* is executed after *command1*
command1 `&&` *command2*	*command2* is executed if *command1* was successful
command1 `||` *command2*	*command2* is executed if *command1* has failed

1.6 Filters

Command	Explanation
`grep` *text filename*	Prints every line in *filename* containing *text*
`sed 's/`*red*`/`*green*`/'` *filename*	Prints every line in *filename* substituting word *red* with word *green*

1.7 Pattern matching

Pattern	Explanation
`*`	matches zero or more characters
`?`	matches one character
`~`	refers to user's default (home) directory

1.8 Miscellaneous

Command	Explanation
`echo` *text*	display a line of text
`history`	view your command line history
`wget` *someWebAddress*	download contents of *someWebAddress* to current directory
`make`	tool that is used to compile source code creating executables
`export` *name*=*value*	sets *value* to *name*. Type `echo $`*name* to view *value*
`source` *filename*	read and execute commands from the *filename* argument

1.9 Syntax

Syntax element	Explanation
\	allows to split command over multiple lines

1.10 `for` loop

`for` loop allows to iterate over a list of items and apply commands on them:

```
for item in item1 item2 item3 item4; do
   echo $item
done
```

Note that we are assigning to `item` each value in list of items (`item1 item2 item3 item4`) in the first line, and we are retrieving the value of `item` by putting $ before it.

We can also write `for` loop in a single line:

```
for item in item1 item2 item3 item4; do echo $item; done
```