Linux/Unix Command Line Cheat Sheet

Command	Description
`pwd`	**p**rints **w**orking **d**irectory, displays the full path of your current location on the filesystem
`ls`	lists contents of current directory
`ls -l`	lists contents of current directory with extra details
`ls /home/user/*.txt`	lists all files in */home/user* ending in .txt
`cd`	change current directory to your home directory
`cd ~`	change current directory to your home directory
`cd /home/user`	change current directory to */home/user*
`cd -`	change current directory to the last directory you were in before your last location
`mkdir mydir`	makes a directory called *mydir*
`rmdir mydir`	removes directory called *mydir*, *mydir* must be empty
`touch myfile`	creates a file called *myfile*. updates the timestamp on the file if it already exists, without modifying its contents
`cp myfile myfile2`	copies *myfile* to *myfile2*. What happens if *myfile2* already exists?
`rm myfile`	removes file called *myfile*
`rm -f myfile`	removes *myfile* without asking you for confirmation. useful if using wildcards to remove files ***
`cp -r dir newdir`	copies the whole directory *dir* to *newdir*, the –r parameter must be specified to copy directory contents recursively
`rm -rf mydir`	this will delete directory *mydir* along with all its content without asking you for confirmation! ***
`nano`	opens the nano text editor, see the status bar at the bottom for help. ^x means CTRL-x. this will exit nano
`nano new.txt`	opens nano editing a file called *new.txt*
`cat new.txt`	displays the entire content of *new.txt* on the terminal
`more new.txt`	displays the content of *new.txt* screen by screen, navigate the file with HJKL or the arrows, press q to quit
`head new.txt`	displays first 10 lines of *new.txt*
`tail new.txt`	displays last 10 lines of *new.txt*
`tail -f new.txt`	displays the contents of *new.txt* as it grows, starting with the last 10 lines, ctrl-c to quit.
`mv myfile newlocdir/`	moves *myfile* into the destination directory *newlocdir*
`mv myfile newfile`	moves *myfile* to *newfile*, effectively renaming it, **if a file called newfile exists, this will overwrite it!**
`mv dir anotherdir`	moves the directory called *dir* to the directory called *anotherdir*, effectively renaming it, if a directory called *anotherdir* already exists, it will simply move the first directory into the second one

`top`	displays all the processes running on the machine, and shows available resources
`du -h`	displays the size of all elements contained in the current directory
`grep` *`pattern files`*	searches for *pattern* in *files*, this command displays the lines in the files containing the pattern
`date`	shows the current date and time
`anycommand` `>` *`myfile`*	redirects the output of *anycommand* writing it to a file called *myfile*, if *myfile* already exists, its previous content will be erased before redirecting the output of *anycommand* into it ***
`anycommand` `>>` *`myfile`*	appends the output of *anycommand* to a file called *myfile*
`tar -zxf` *`archive.tgz`*	extracts the contents of the archive called *archive.tgz* ***
`tar -zcf` *`dir.tgz dir`*	creates a compressed archive *called dir.tgz* containing all the files and directory of *dir*
`time` *`anycommand`*	executes *anycommand* and displays the it took the OS to run it
`man` *`anycommand`*	displays the **man**ual of *anycommand*
`cal -y`	displays the calendar of the current month by default, add the -y parameter to get the whole year
`CTRL-c`	kills whatever process you're currently executing
`CTRL-insert`	copies selected text to the clipboard (n.b. see above, ctrl-c will kill whatever you're doing)
`SHIFT-insert`	pastes clipboard contents to terminal

*** = use with extreme caution! you can easily delete or overwrite important files with these.

Absolute vs relative paths.

Let's say you are here that your current directory is: */home/student/scripts/*

If you wanted to go to */home/student/*, you could type: `cd /home/student/`.

Or you could use a relative path: `cd ..` (two periods). This will take you one directory "up" of the current directory, effectively to the parent directory of the current directory.

`.`	(a single period) means the current directory
`..`	(two periods) means the parent directory
`~`	means your home directory

A few examples

`mv myfile ..`	moves *myfile* to the parent directory
`cp myfile ../newname`	copies *myfile* to the parent directory and names the copy newname
`cp /home/student/scripts/life.c .`	copies *life.c* to ".", meaning the current directory you're in
`cp myfile ~/subdir/newname`	copies *myfile* to *subdir* in your home directory and naming the copied file *newname*
`more ../../../myfile`	displays screen by screen the content of *myfile*, which exists 3 directories "up"

Wildcards (use carefully, especially with rm)

`*` matches any character. example: `ls *.pl` lists any file ending with ".pl" ; `rm dataset*` will remove all files beginning with "dataset"

`[xyz]` matches any character in the brackets (x, y, or z). example: `cat do[or]m.txt` will display the contents of either doom.txt or dorm.txt