Beginner’s Linux Cheat Sheets

February 2009 - Andrea Minoia

This document wants to be a first aid for new Linux user. A google search will produce a huge number of link to cheat sheets, tutorial, guide and often is difficult to find that basic command you need. This is not a tutorial nor a guide and is far to be a complete linux user reference. Is just a starting point, something you can use when you are sit in front of your terminal and you have no clue on what’s next.

I have decided to introduce four topics: bash, vi/vim, gnuplot and queuing systems on remote clusters. This because this collection of basic cheat sheets has been written to help visitors coming in our laboratory for a short staying. In particular for those who are not familiar with linux and computational chemistry.

List of topics:

Beginner's Linux Cheat Sheets

Bash's basic cheat sheet

mkdir - make a directory

make *dir*	: create dir in the current dir

cd - change dir

cd ~ / cd	: go to home dir
cd *path/of/the/dir/u/want/to/go*	: move to the specified directory
cd ..	: go to the parent directory

rm - remove

rm *file*	: delete file
rm -r *dir*	: delete directory
rm -f	: delete without asking confirmation

mv - move / rename

mv *file newfile*	: rename file in newfile
mv *file new/path*	: move file in the specified dir
mv *file new/path/newfile*	: move file as newfile in the specified dir
mv *-f*	: move without confirmation if overwriting

cp - copy

cp *file newfile*	: copy file in newfile
cp *file new/path*	: copy file in the specified dir
cp *file new/path/newfile*	: copy file as newfile in the specified dir
cp *-f*	: copy without confirmation if overwriting

ls - list file

ls	: list content current dir
ls -l	: list content current dir in a detailed way
ls -a	: list hidden files in the current dir
ls */other/path*	: fist content of another dir
ls *file*	: check if file is present in the current dir

more / cat / tail - display content file

more *file*	: display content of file bit by bit
cat *file*	: display content of file all at once
tail *file*	: display last 10 lines of file
tail -n 20 *file*	: display last 20 lines of file
tail -f *file*	: display last 10 lines of file each time file is updated

> < | - operators

cat file > *newfile*	: redirect output of cat to newfile (overwrite if newfile exists, otherwise newfile is created)
cat file >> *newfile*	: append output of cat to newfile
command < *file*	: redirect file into a command (e.g. a program)
grep *string* | more	: use the output of grep as input of command more

? * - wildcards

rm -rf *	: remove all files and dirs in the current dir
mv file* */new/path*	: move all files whose name starts for file in new/path
rm -f *file_00?.com*	: ? stays for every character in that position

tar / zip / gzip - archives

zip/unzip *file.zip*	: create/extract compressed zip archives
gzip/gunzip *file.gz*	: create/extract compressed gzip archives
tar cvf *file.tar **	: create non-compressed tar archived named file.tar of all the files and dirs in the current dir
tar xvf *file.tar*	: extract tar archives
tar czvf *file.tar.gz (or .tgz)*	: create compressed tar archives (same as gzip file.tar)

tar xzvf *file.tar.gz* : extract compressed tar archives (same as gunzip file.tar.gz)

ssh / scp - connect to remote server (all commands given from local server)

ssh *user@server* : connect to server (server can be the name or the IP address)
ssh -XY *user@server* : allows export graphical display
scp *file user@server:/path* : copy file from local to remote server
scp -r *dir user@server:/path* : copy dir from local to remote server
scp *user@server:/path/file .* : copy file from remote server to local dir

grep - find a string in files

grep *"string_to_search" file* : search the string in file
grep -R *"string_to_search" ** : search the string in all files and directories
grep -c "string" file l nl : search string in file and return the number of lines containing string. The output of grep is suppressed by the flag -c

df - disk usage

df -m : display disk usage in MB

sed - stream editor

sed 's/day/night/' <old >new : search all words "day" in file *old* and replace them with "night" in the new file *new*

paste - paste two file

paste *file1 file2 > out* : paste *file1* and *file2*. Usefule to create multicolumn file by merging two input files.

man - get help

man *command* : display help for command

Bash: advanced use

The bash shell is not only a place where type commands, but is a powerful and flexible environment that allow you to create scripts and programs. Here some examples:

for /do /done - do something over a loop

for f in scan.*; do analyze $f -k file.key e >> anen.out ; done
: perform a tinker energy analysis on all files scan.* and append the results in anen.out

useful combination of commands commands:

cp $(ls l egrep -v '^#') /where/to/copy : copy all files except those starting with # in the specified location

&, ctrl+z, bg and fg - run in background

molden & : start and run molden in background
ctrl+z : suspend a running program
bg : send in background a suspended program
fg : bring in foreground a program running in bg

&& - and

make all && *make rename* : execute two commands one after the other.

ctrl+t - flip last two characters typed

teh ctrl+t : you get *the*

Return to the list of topics

VI /VIM's cheat sheet

vi is a widely used text editor, extremely powerful but usually lack of a graphical interface, which make things a little bit complicate for new user. There are two "operating modes" in vi: **edit mode** and **command mode**. When vi starts, you are in command mode: type i to go in edit mode and start to write. Press the escape key to return in command mode. Please, forget your mouse ;)

from command mode you can go to insert mode with:

- i : start editing from the current position of the cursor
- a : start editing from the position next to the current position of the cursor
- shift+a : start editing from the end of the line
- o : insert a newline below the current line, and start editing
- shift+o : insert a newline above the current line, and start editing

some commands in command mode:

- x : delete current character
- *n* dd : delete *n* lines starting from (and including) the current one (dd to delete 1line)
- shift+g : jump to the end of the file
- *n*+shift+g : jump to the *nth* line
- *1*+shift+g : jump to the 1st line
- ctrl+g : info about file (name, current line, total line, ...)
- *n* shift+y : yank *n* lines starting from (and including) the current one (Y to yank 1 line)
- p : past yanked or deleted lines below the current line
- shift+p : past yanked or deleted lines above the current line
- . : repeat last command or text entry
- u : undo last command
- /*string* : search *string* in file. type n to jump to the next occurrence, shift+n to jump to the previous occurrence
- :1,$s/*string*/*string2*/ : replace *string1* with *string2* in all the file
- :*n*,*m*s /*string*/*string2*/ : replace *string1* with *string2* from line *n* to line *m*
- :sp : split horizontally the current vi windows in two
- :vsp : split vertically the current vi windows in two
- ctrl+w up/down/left/right : change active window (if window has been split)
- :e *file* : open a file /create new file

save, quit, open (command mode):

- :w : write (save)
- :w! : overwrite an existing file
- :q : quit
- :q! : quit without saving
- :wq : save and quit

visual block (command mode)

- ctrl+v : select part of a file (move with the cursor) and then you can use dd, Y, x, ...

Ask google to tell you more about vi, or ask man, or have a look into the doc directory of your crunchvm.

Return to the list of topics