COMP1005 Linux Vim Cheat Sheet

Lisa Luff

8/12/2020

Contents

Unix/Terminal/Shell

Unix Help

- Help and information documents run a program called pager
  - You need to use key binding to move around pager documents
    - Move down: j, down, page down, enter, or space
    - Move up: k, up, or page up
    - Quit: q

Help Commands

Command	Purpose
man *some_command*	Read the manual page for *some_command*
info *some_command*	Read a more detailed manual (mostly only for GNU programs)
apropos *'search term'*	Find a command whose man page has a matching summary
whatis *some_command*	Print the summary from the man page for *some_command*
whereis *some_command*	Print the path to *some_command* and other paths significant to it

File System

- / is used to direct to a file
  - You will need to work your way down from the top (files are hierarchial), so it might be /top-file/middlefile/actualfile, etc. to be in the file you want
- To delete a file, you need to delete the file, and have no processes have a file descriptor for it
- . refers to the file you are in, and .. refers to the parent file (or file that file is in)

Working with the file system

Command	Purpose
cd *path*	Change your working directory to *path*
ls -*options path*	List files within directory at *path*
pwd	Print working (current) directory
history *n*	Lists *n* most recent commands, can save to file (use -f to search)
cp -*options source dest*	Copy the file at *source* to *destination* file
mv -*options source dest*	Move the file at *source* to *destination* file
rm -*options path*	Remove the file at *path* (use -f to search)
mkdir -*options path*	Create a new directory at *path*
rmdir -*options path*	Remove directory (if empty, or use -R to delete it and contents)
chmod -*options mode path*	Change the permissions of *path* to *mode* (mode is who has permission)
. *path*	Execute *path*
-la	An *option* for all above, shows hidden files
-R	An *option* for above to enact on subdirectories as well

Working With Processes

Command	Purpose
pgreg -*options pattern*	Print the IDs of processes that match *pattern*
ps -*options*	Print a list of running processes and some details about them
top -*options*	Monitor a list of running processes and some details about them
kill -*signal pid*	Send *signal*(type of kill) to a process with the ID *pid*(process to end)
fg *pid*	Continue a recently stopped process *pid* in the foreground
bg *pid*	Continue a recently stopped process *pid* in the background
ctrl q	Will run terminal again if you accidentally press ctrl s

Working With Text and Other Data

Command	Purpose
vim *file*	Edit the given *file* using vim
echo *words/variable*	Print the given *words/variable*
less *path*	Display the contents of the *path* in a pager (alternative: more, different navigation of page)
head -*n path*	Print first 10 (or *n*) lines of *path*
tail -*n path*	Print last 10 (or *n*) lines of *path*
more *path*	Open and scroll through *path* one page at a time (space or 'n' - continue, 'b' - backwards, 'q' - exit)
less *path*	Free and open source version of more
awk - F "*sep*" '{*do a, b, c*}' *path*	Reads file, separates with *sep*, counts from 1, prints *a, b, c* - as columns
wget *website*	Get files from web
wc *path*	Word count, three numbers are lines, words, characters
sed s/*from*/*to*/*option*	Print the input, with text matching *from* changed to *to* (use g as the *option* to make it global for the file)
diff *variable/file1 variable/file2*	Print only the lines that are different between *1* and *2*
gnuplot -p *path*	Allows command line plotting of gnuplot in *path*, need package, and script

Grep

- Allows sophisticated searches using regular expressions with commands

Command	Purpose
grep *option pattern(string) path*	Print only the lines in *path* that match *pattern* - as lines
path	starts with / and can use ../ to look in parent directory
(*a*|*b*)	For multiple arguments
command path | grep *pattern*	To use commands with grep
-r or -R will look in directory and all subdirectories	
-i	Not case sensitive
-c	Counts occurances
-w	Will find only exact
-W	Same but for words
-n	Line number found on
-B	Line before found
-A	Line after found
-h	Suppress file names
-*x**y*	How to combine options
–color *option*	Can change colours of output
- P	An option that allows the use of regular expressions
:———	:———
- P command	Use
:———	:———
-v	As option for inverse
.thing	Can be 0 or 1 *things*
.*	Wild card
.{*n*, *m*}	Can be between *n* and *m* random assortment of things
^	Start of line
$	End of line

Combining Commands

Command	Purpose
variable/file1 | *variable/file2*	Take output of *1*, and pass into *2* as input
variable/file1 > *variable/file2*	Take output of *1*, and send to *2* instead of terminal
cat *path*	Concatenate the given files, while printing their contents
cat *variable/file1* » *variable/file2*	Append *1* to *2*
variable/file1 < *variable/file2*	Execute *1*, but take input from *2* instead of terminal

Bash Scripts

- **Note**: Spaces are important! Cannot have any extra white space
- Create with vim *filename*.sh - .sh is the file type for all bash scripts (sh for shell)
- Run with sh *filename*.sh
- Uses the same language as terminal
- Can only comment with #
- Use command line arguments:
  - Don't need to download anything
  - In script set variables to be defined as $*n*
    - $1 - is the command to run the script
    - $2 - onwards - is the arguments after 1, all separated with spaces
    - $@ - is all command line arguments
- You can use command line functions in the script exactly as you would in the command line
  - Can use mkdir, cd, etc.
- echo is print, don't need brackets or anything, just not whatever separated with spaces
- Use functions with accents ‘*function*‘
  - Range: ‘seq *low step high*‘
  - Date: ‘date “+%Y-%M-%d_%H:%M:%s”‘
- For loop:
  - for *item* in $*range*
  - do
  - *thing*
  - done

Create a Package

- Modules are the programs containing functions
- Packages are the directory containing the modules
  - Makes a directory a package using:
    - ___init___.py in the directory
  - Call as import *package.module*

Creating a Global Package

- Add to PyPI to use with pip install
- PyPI - https:////pypi.python.org/
- Need to consider structure
  - Need to use guides beyond PEP8
  - Eg. PEP257 - docstring conventions
- Package name constraints:
  - All lowercase
  - Unique on pypi, even if you don't want to make your package publicly available
  - Underscore-separated or no work separators at all (no hyphens)
- Directory structure
  - Top level directory is the root of the SCN repo, eg *package*.git
  - Sub-directory of the same name is the actual python module, holds:
    - ___init___.py
    - setup.py -
      from setuptools import setup
      setup(name = '*package*',
      version = '*n*',
      description = '*string*',
      url = '*url*',
      author = '*me*',
      author_email = '*email*',
      license = 'MIT',
      packages = ['*package*'],
      zip_safe = False)
      - Then the package can be downloaded with pip locally
- To register the package to PyPI
  $ python setup.py register
  - If you haven't registered anything before you will need an account
  - Then anyone can download it with pip, and it can be made a dependency of other packages, and be automatically installed when that package is installed
- Ignoring files
  - Don't want to include all files in the package (Eg. intermediaary files made automatically by Python during development)
  - Use .gitignore to automate (or equivalent for other SCM/VCS's)
    - Compiled Python modules:
      *.pyc
    - Setuptools distribution folder:
      /dist/
    - Python egg metadata, regenerated from source by setuptools
      /*.egg-info

Creating and Opening Zip Files

- Create the zip file:
  - zip *filename* *
  - This creates a zip file containing everything in your current directory
- To save with a specific file name:
  - zip -R *chosenname* '*nameofcurrentfile*' *
- See the contents inside a zip file
  - unzip -1 *filename*.zip

Accessing Jupyter Notebooks

- Type jupyter notebooks into terminal in the directory you want to work in
- Use control-c to close the jupyter notebooks

Git Version Tracking

- Three trees of Git:
  - The HEAD - last commit snapshot, next parent
  - Index - Proposed next commit snapshot
  - Working directory - Sandbox
- Basic workflow:
  - Init a repo (possible init of clone)
    $ git init
  - Tell git who you are
    $ git config –global user.name "*your name*"
    $ git config –global user.email "*your email*"
  - Edit files
  - Stage the changes
    $ git status
  - Review your changes
    $ git add *filename*
    $ git status
  - Commit the changes of directory with a comment
    $ git commit -m "*comment*"
- Checking changes and history:
  - git diff - Show the difference between working directory and staged
  - git diff –cached - Show the difference between staged and the HEAD
  - git log - View history
- Using backups:
  - git checkout *commit hash*
    * Commit hash is the first 4 numbers of commit when looking at the log
- Using remote repository:
  - Get changes
    * git fetch
    * git pull (fetches and merges)
  - Propagate changes
    * git push
  - Protocols
    * Local filesystem - (file:////)
    * SSH - (ssh:////)
    * HTTP - (http://// or https:////)
    * Git protocol (git:////)

makeuseof.com

Unix/Linux Command Reference

File commands

ls	Directory listing
ls -al	Formatted listing with hidden files
cd dir	Change directory to dir
cd	Change to home
pwd	Show current directory
mkdir dir	Create a directory dir
rm file	Delete file
rm –r dir	Delete directory dir
rm –f file	Force remove fire
rm –rf dir	For remove directory dir
cp file1 file2	Copy file1 to file2
cp –r dir1 dir2	Copy dir1 to dir2; create dir2 if it doesn't exit
mv file1 file2	Rename or move file1 to file2. If filed 2 is an existing directory, moves file1 into directory file2
ln –s file link	Create symbolic link link to file
touch file	Create or update file
cat > file	Places standard input into file
more file	Output the contents of file
head file	Output the firest 10 lines of file
tail file	Output the last 10 lines of file
tail –f file	Output the contents of file as it grows, starting with the last 10 lines

Process Management

ps	display all currently active processes
top	display all running processes
kill pid	kill process id pid
killall proc	kill all processes named proc *
bg	lists stopped or background jobs; resume a stopped job in the background
fg	Brings the most recent job to the foreground
fg a	brings job a to the foreground

File Permissions

chmod octal file	change the permissions of file to octal, which can be found separately for user, group, and world by adding: • 4 – read (r) • 2 – write (w) • 1 – execute (x) **Examples:** ***chmod 777*** *– read, write, execute for all* ***chmod 755*** *– rwx for owner, rx for group and world. For more options, see* **man chmod.**

SSH

ssh user@host	connect to host as user
ssh -p port user@host	connect to host on port port as user
ssh-copy-id user@host	add your key to host for user to enable a keyed or passwordless login

Searching

grep pattern files	search for pattern in files
grep -r pattern dir	search recursively for pattern in dir
command | grep pattern	search for pattern in the output of command
locate file	find all instances of file

System Info

date	show the current date and time
cal	show this month's calendar
uptime	show current uptime
w	display who is online
whoami	who you are logged in as
finger user	display information about user
uname -a	show kernel information
cat /proc /cpuinfo	cpu information
cat /proc /meminfo	memory information
man command	show the manual for command
df	show disk usage
du	show directory space usage
free	show memory and swap usage
whereis app	show possible locations of app
which app	show which app will be run by default

Compression

tar cf file.tar files	create a tar named file.tar containing files
tar xf file.tar	extract the files from file.tar
tar czf file.tar.gz files	create a tar with Gzip compression
tar xzf file.tar.gz	extract a tar using Gzip
tar cjf file.tar.bz2	create a tar with Bzip2 compression
tar xjf file.tar.bz2	extract a tar using Bzip2
gzip file	compresses file and renames it to file.gz
gzip -d file.gz	decompresses file.gz back to file

Network

ping host	ping host and output results
whois domain	get whois information for domain
dig domain	get DNS information for domain
dig -x host	reverse lookup host
wget file	download file
wget -c file	continue a stopped download

Installation

Install from source:	
./configure	
make	
make install	
dpkg -i pkg.deb	install a package (Debian)
rpm -Uvh pkg.rpm	install a package (RPM)

Shortcuts

Ctrl+C	halts the current command
Ctrl+Z	stops the current command, resume with fg in the foreground or bg in the background
Ctrl+D	log out of current session, similar to exit
Ctrl+W	erases one word in the current line
Ctrl+U	erases the whole line
Ctrl+R	type to bring up a recent command
!!	repeats the last command
exit	log out of current session
*	use with extreme caution

Vi/Vim

- To open Vim, type vim and the name of the file you want to edit

Command Mode

- Command mode - Can use editing commands to manipulate text

Enter insert mode:

Command	Purpose
a	Append. Text is inserted immediately after the cursor position, on the same line
i	Insert. Text is inserted before the position of the cursor position, on the same line
o	Open below. Text is inserted on a new line immediately below the postition of the cursor.
O	Open above. Text is inserted on a new line immediately above the position of the cursor

Cursor movement

Command	Purpose
j	Up
k	Down
h	Left
l	Right
Cursor Keys	Instead of j, k, h, l
w	Move to beginning of the next word
e	Move to the end of the word
0	Move to the beginning of the current line
$	Move to the end of the current line
G	Move to the last line in the file
nG or :n	Move to line n
Ctrl + u	Page up
Ctrl + d	Page down

Editing text:

Actions (x)	Purpose
x	Delete character at the position of the cursor
cy	c for change, will delete and enter insert mode
dy	d for delete
yy	y for yank, copy
py	p for paste
Acts on (y)	**Purpose**
xe	Enact shortcut to the end of word
xb	Enact shortcut back to beginnging of word
xw	Enact shortcut to next word
xx	Repeat shortcut twice to enact on entire line
uppercase	Change shortcut to upper case enact it to the end of the line
xG	Do thing to end of document
Additional acting specifiers	**Purpose**
nxy	Enacts shortcut n number of times
xiy	i for in, enacts shortcut on current position
xtz	t for till, enacts shortcut until character specified by z
Other actions	**Purpose**
u	Undo the action of the previous command
.	Redo the last command

Search Text

Command	Purpose
/*string* - enter	Search forward
?*string* - enter	Search backward
n	Find next occurrence
N	Find previous occurrence

Insert Mode

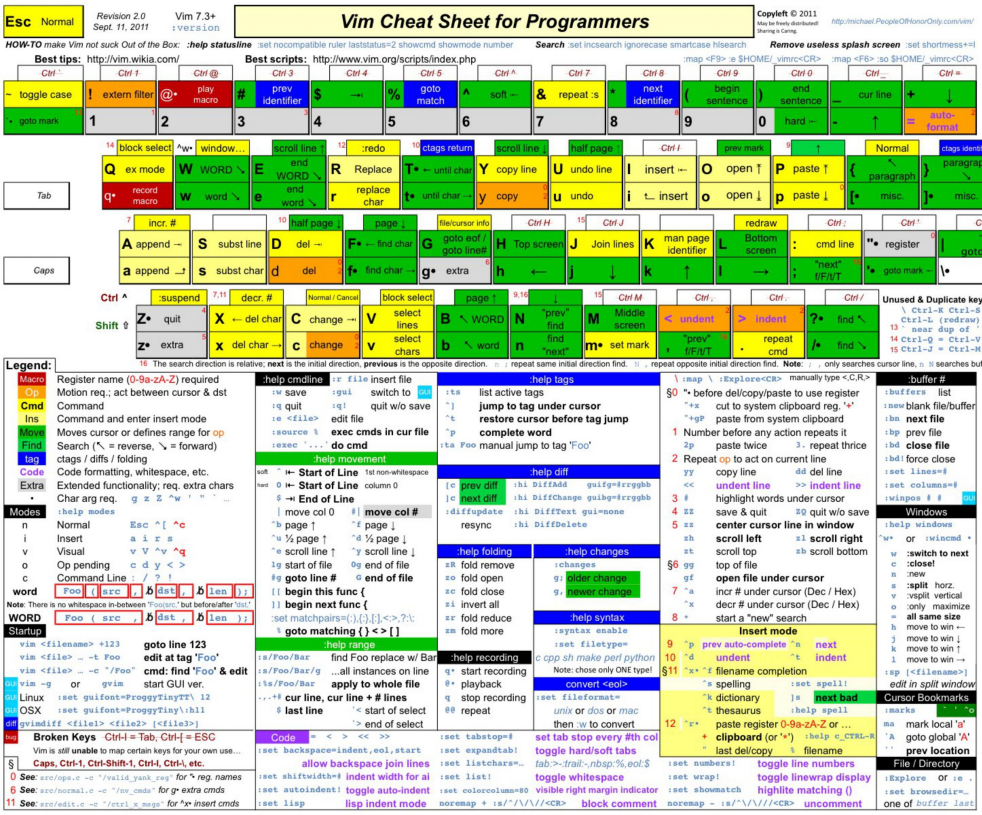
- Insert mode - Can enter text into file being edited
- Esc - Takes you back to command mode

Exiting Vim:

Command	Purpose
ZZ OR :wq	Save and exit
:w	Save do not exit
:w *filename*	Changes the name the of the file
:q	Dot not save, exit only if no changes were made
:q!	Exit but do not save

Retrieving lost files

- **ctrl z** will close vim, but what you were working on will still run in the background, even if unsaved
- It will be held in a temporary file *.filename.filetype*.swp


Esc Normal
Revision 2.0 Sept. 11, 2011
Vim 7.3+ :version
Vim Cheat Sheet for Programmers
Copyleft © 2011
http://michael.PeopleOfHonorOnly.com/vim/
HOW-TO make Vim not suck Out of the Box: :help statusline :set nocompatible ruler laststatus=2 showcmd showmode number
Search :set incsearch ignorecase smartcase hlsearch
Remove useless splash screen :set shortmess+=I
Best tips: http://vim.wikia.com/
Best scripts: http://www.vim.org/scripts/index.php
:map <F9> :e $HOME/_vimrc<CR>
:map <F6> :so $HOME/_vimrc<CR>
Ctrl ` toggle case goto mark
Ctrl 1 extern filter
Ctrl @ play macro
Ctrl 3 prev identifier
Ctrl 4 →
Ctrl 5 goto match
Ctrl ^ soft ←
Ctrl 7 repeat :s
Ctrl 8 next identifier
Ctrl 9 begin sentence
Ctrl 0 end sentence hard ←
Ctrl _ cur line ← ↑
Ctrl = ↓ auto-format
Tab
block select ^w• window…
scroll line ↑ :redo
ctags return
scroll line ↓
half page ↑
Ctrl I
prev mark
↑
Normal
ctags identifier
Q ex mode
W WORD ↘
E end WORD ↘
R Replace
T• ← until char
Y copy line
U undo line
I insert ↤
O open ↑
P paste ↑
{ ↖ paragraph
} paragraph ↘
q• record macro
w word ↘
e end word ↘
r replace char
t• until char →
y copy
u undo
i ↳ insert
o open ↓
p paste ↓
[• misc.
]• misc.
Caps
incr. #
half page ↓
page ↓
file/cursor info
Ctrl H
Ctrl J
redraw
Ctrl ;
Ctrl '
Ctrl \
A append →
S subst line
D del →
F• ← find char
G goto eof / goto line#
H Top screen
J Join lines
K man page identifier
L Bottom screen
: cmd line
"• register
| goto col#
a append ↲
s subst char
d del
f• find char →
g• extra
h ←
j ↓
k ↑
l →
; "next" f/F/t/T
'• goto mark →
\•
Ctrl ^
:suspend
Normal / Cancel
block select
page ↑
Ctrl M
Ctrl <
Ctrl >
Ctrl /
Shift ⇧
Z• quit
X ← del char
C change →
V select lines
B ↖ WORD
N "prev" find
M Middle screen
< undent
> indent
?• find ↖
z• extra
x del char →
c change
v select chars
b ↖ word
n find "next"
m• set mark
. repeat cmd
/• find ↘
Unused & Duplicate keys
\ Ctrl-K Ctrl-S (free)
Ctrl-L (redraw)
near dup of
Ctrl-Q = Ctrl-V
Ctrl-J = Ctrl-M = ^N
Legend:
Macro Register name (0-9a-zA-Z) required
Op Motion req.; act between cursor & dst
Cmd Command
Ins Command and enter insert mode
Move Moves cursor or defines range for op
Find Search (↖ = reverse, ↘ = forward)
tag ctags / diffs / folding
Code Code formatting, whitespace, etc.
Extra Extended functionality; req. extra chars
• Char arg req. g z Z ^w ' " ` …
Modes :help modes
n Normal Esc ^[^c
i Insert a i r s
v Visual v V ^v ^q
o Op pending c d y < >
c Command Line : / ? !
word Foo (src , ß dst , ß len);
WORD Foo (src , ß dst , ß len);
Note: There is no whitespace in-between 'Foo(src,' but before/after 'dst,'
Startup
vim <filename> +123 goto line 123
vim <file> … -t Foo edit at tag 'Foo'
vim <file> … -c '/Foo' cmd: find 'Foo' & edit
vim -g or gvim start GUI ver.
Linux :set guifont=ProggyTinyTT\ 12
OSX :set guifont=ProggyTiny\:h11
gvimdiff <file1> <file2> [<file3>]
bug Broken Keys Ctrl-I = Tab; Ctrl-[= ESC
Vim is still unable to map certain keys for your own use…
§ Caps, Ctrl-1, Ctrl-Shift-1, Ctrl-I, Ctrl-\, etc.
0 See: src/opa.c -c "/valid_yank_reg" for "• reg. names
6 See: src/normal.c -c "/nv_cmds" for g• extra cmds
11 See: src/edit.c -c "/ctrl_x_msgs" for ^x• insert cmds
:help cmdline
:r file insert file
:w save :gui switch to GUI
:q quit :q! quit w/o save
:e <file> edit file
:source % exec cmds in cur file
:exec '...' do cmd
:help movement
soft ^ ⇤ Start of Line 1st non-whitespace
hard 0 ⇤ Start of Line column 0
$ ⇥ End of Line
| move col 0 #| move col #
^b page ↑ ^f page ↓
^u ½ page ↑ ^d ½ page ↓
^e scroll line ↑ ^y scroll line ↓
1g start of file 0g end of file
#g goto line # G end of file
[[begin this func {
]] begin next func {
:set matchpairs=(:),{:},[:],<:>,?:\;
% goto matching { } < > [ ]
:help range
:s/Foo/Bar find Foo replace w/ Bar
:s/Foo/Bar/g …all instances on line
:%s/Foo/Bar apply to whole file
...+# cur line, cur line + # lines
$ last line 'x start of select
'> end of select
:help tags
:ts list active tags
^] jump to tag under cursor
^t restore cursor before tag jump
^p complete word
:ta Foo manual jump to tag 'Foo'
:help diff
]c prev diff :hi DiffAdd guifg=#rrggbb
[c next diff :hi DiffChange guibg=#rrggbb
:diffupdate :hi DiffText gui=none
resync :hi DiffDelete
:help folding
zR fold remove
zo fold open
zc fold close
zi invert all
zr fold reduce
zm fold more
:help changes
:changes
g; older change
g, newer change
:help recording
q• start recording
@• playback
q stop recording
@@ repeat
:help syntax
:syntax enable
:set filetype=
c cpp sh make perl python
Note: chose only ONE type!
convert <eol>
:set fileformat=
unix or dos or mac
then :w to convert
\ :map \ :Explore<CR> manually type <,C,R,>
§0 "• before del/copy/paste to use register
"+x cut to system clipboard reg. '+'
"+gP paste from system clipboard
1 Number before any action repeats it
2p paste twice 3. repeat thrice
2 Repeat op to act on current line
yy copy line dd del line
<< undent line >> indent line
3 # highlight words under cursor
4 ZZ save & quit ZQ quit w/o save
5 zz center cursor line in window
zh scroll left zl scroll right
zt scroll top zb scroll bottom
§6 gg top of file
gf open file under cursor
7 ^a incr # under cursor (Dec / Hex)
^x decr # under cursor (Dec / Hex)
8 * start a "new" search
Insert mode
9 ^p prev auto-complete ^n next
10 ^d undent ^t indent
§11 ^x^ ^f filename completion
^s spelling :set spell!
^k dictionary]s next bad
^t thesaurus :help spell
12 ^r• paste register 0-9a-zA-Z or …
+ clipboard (or '+') :help c_CTRL-R
" last del/copy % filename
:buffers # list
:new blank file/buffer
:bn next file
:bp prev file
:bd close file
:bd! force close
:set lines=#
:set columns=#
:winpos # #
Windows
:help windows
^W• or :wincmd •
w :switch to next
c :close!
n :new
s :split horz.
v :vsplit vertical
o :only maximize
= all same size
h move to win ←
j move to win ↓
k move to win ↑
l move to win →
:sp [<filename>]
edit in split window
Cursor Bookmarks
:marks
ma mark local 'a'
'A goto global 'A'
'' prev location
File / Directory
:Explore or :e .
:set browsedir=…
one of buffer last
Code
= < > << >>
:set backspace=indent,eol,start allow backspace join lines
:set shiftwidth=# indent width for ai
:set autoindent! toggle auto-indent
:set lisp lisp indent mode
:set tabstop=# set tab stop every #th col
:set expandtab! toggle hard/soft tabs
:set listchars=… tab:>-,trail:~,nbsp:%,eol:$
:set list! toggle whitespace
:set colorcolumn=80 visible right margin indicator
noremap + :s/^\/\//<CR> block comment
:set numbers! toggle line numbers
:set wrap! toggle linewrap display
:set showmatch highlite matching ()
noremap - :s/^\/\///<CR> uncomment