CS143A
Principles on Operating Systems
Discussion 06:

Instructor: Prof. Anton Burtsev

TA: Saehanseul Yi (Hans)

Nov 8, 2019 **2 PM**

Agenda

- Useful vim shortcuts
- tmux: terminal multiplexer
- ctags: generate an index file of names found in source code
- cscope: search source code

VIM Shortcuts

- Recap: normal mode vs. insert mode
- Navigation: `h`← `j`↓ `k`↑ `l`→
  - Pros: you don’t have to move your hand to the arrow keys
  - it takes some time to get used to it. It is totally fine to use arrow keys
- Page up/down: Ctrl+f/b
- Go to the beginning/end of the file: gg/G
- Current line in the center of screen: zz
- Go to line: :<number>

VIM Shortcuts

- insert after the focus: a (vs. i) (will go into insert mode)
- add a new line below/up: o/O (will go into insert mode)
- undo/redo: u/Ctrl+r
- go to start/end of the line: ^/$
- delete a line/word/character: dd/dw/x
- copy a line: yy
- paste: p
- select lines/block: v/Ctrl+v (then usually 'y' to copy or 'x' to delete)

VIM Shortcuts

- replace one character/continuously: r/R
- replace a word: cw
- select a word and highlight: *
- No highlights: :noh
- move to next/previous highlighted keword: n/N
- search: /
- repeat commands: <any number> + shortcut

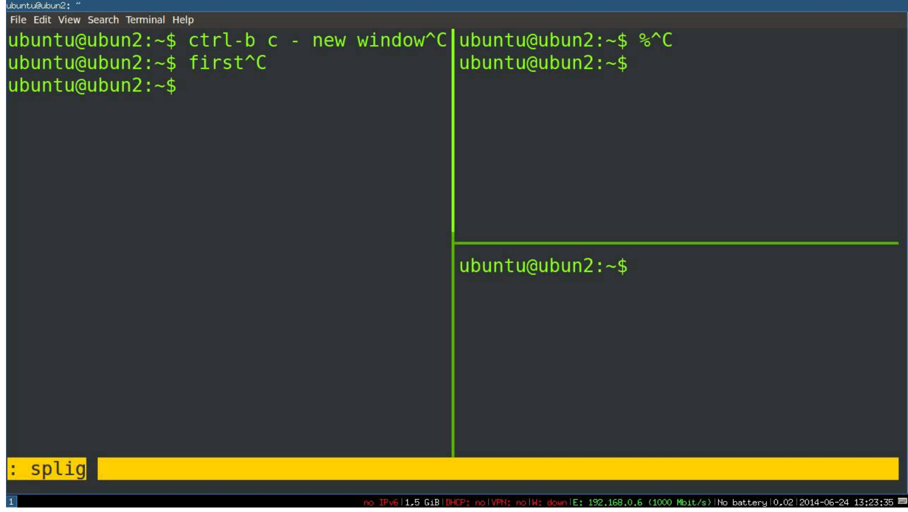
VIM Configuration

- **~/.vimrc** (‘~’ represents your home directory, i.e. /home/saehansy)
- below configs can be dynamically applied at runtime using **:**
  (e.g. :set nu) (to cancel, usually put ! at the end. e.g. :set nu!)
- **set nu**: line numbers
- **set smartindent**: when you add a new line, it automatically indents
- **set softtabstop=4**
  set tabstop=4
  set shiftwidth=4
- **set expandtab**: insert spaces instead of tab
- **set hlsearch**: highlight keyword

VIM Shortcuts

- Align source code: =
- Increase/decrease indent of a line: >>/<<
- split window vertically/horizontally: Ctrl+w, v/s
- Move between windows: Ctrl+w, hjkl(arrow keys also work)
- VIM can open multiple files
  :buffers
- To select buffer: :buffer <buffer_num>

TMUX: Terminal Multiplexer



```
ubuntu@ubun2:~$ ctrl-b c - new window^C│ubuntu@ubun2:~$ %^C
ubuntu@ubun2:~$ first^C                │ubuntu@ubun2:~$
ubuntu@ubun2:~$                        │
                                       ├──────────────────────
                                       │ubuntu@ubun2:~$
: splig
1          no IPv6|1,5 GiB|DHCP: no|VPN: no|W: down|E: 192.168.0.6 (1000 Mbit/s)|No battery|0,02|2014-06-24 13:23:35
```

- Run multiple terminals in one window
- Resides in servers, saves the state you're working on
  - However, openlab machines are shared by many people
  - When you don't use it, please quit tmux for others
- Already installed in openlab machines
- type tmux or tmux a
- In tmux, everything revolves around 'prefix' (default: Ctrl + b)
- To quit, prefix x then y or just type quit to exit your session
- Conf file: ~/.tmux.conf

Basic TMUX configuration

- ~~unbind C-b~~
  ~~set -g prefix C-a~~
  ~~bind C-a send-prefix~~
- bind C-q kill-window
- bind | split-window -h
- bind - split-window -v
- setw -g mode-mouse on
  set -g mouse-select-pane on
  set -g mouse-resize-pane on
  set -g mouse-select-window on
  set -g mouse-utf8 on

C- means Ctrl-

Changing prefix. I found C-a convenient for me.
You can use default(C-b) or something else
To send the program C-a(like xv6), press C-a C-a

Quickly quit the window

Splitting window horizontally or vertically

You can use mouse to select a window,
resize the pane, and scroll

Basic TMUX commands

- prefix c : new window
- prefix n/p: switch to next/previous window
- prefix |(pipe) / - : split window vertically or horizontally
- prefix ← ↓ ↑ →: move between pane(split window)
- prefix q: kill the window
- prefix x: kill the pane
- prefix ,: rename window

Having trouble quitting xv6?

- open another terminal, and type
  killall qemu-system-i386

Ctags: Navigate code like a pro

- index names(functions, variables, …) into a file (tags)
- Unfortunately, openlab machines doesn't have ctags
  (We cannot use 'apt install …' because we are not superusers)
- Build from source code!
- git clone https://github.com/universal-ctags/ctags
- mkdir ~/local
  cd ctags/
  ./autogen.sh
  ./configure --prefix=/home/<UCNetID>/local
  make
  make install

For macOS users,
install brew, the package manager
(https://brew.sh)
brew install ctags

configure – make – make install

For Windows Linux Subsystem users,
sudo apt install ctags

Ctags: Navigate code like a pro

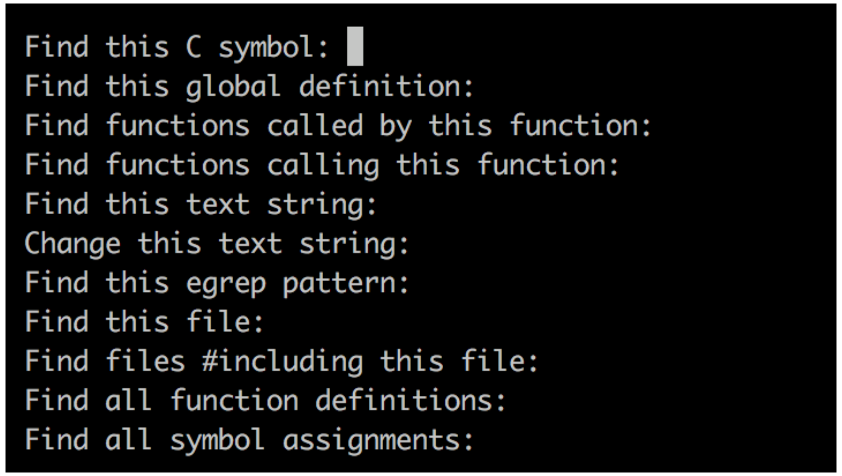
- Now ctags is installed in /home/<UCNetID>/local/bin
- Add this path to PATH in .bashrc(or .bash_profile)
  export PATH=$HOME/local/bin:$PATH

Ctags: Navigate code like a pro

- index names(functions, variables, …) into a file (tags)
- In the source code directory, run ‘ctags -R’
- (In vim) Place your cursor to target name, press Ctrl + [, go back Ctrl + T
- *DISCLAIMER: it is not perfect, it doesn’t understand complex syntax*
  But most of the time, it works pretty well
- C MACRO: to the definition
- variable: to the definition
- function: to the definition

Cscope: A Powerful code-searching tool

- Works outside vim (it invokes vim)
- Find all the C symbol references, definitions, calling/called reference, ...
- Also finds some arbitrary texts including comments
- Not installed in openlab machines..
- git clone https://github.com/portante/cscope
- **./configure --prefix=/home/<UCNetID>/local**
  make
  make install



```
Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find all function definitions:
Find all symbol assignments:
```

For macOS users,
install brew, the package manager
(https://brew.sh)
brew install cscope

For Windows Linux Subsystem users,
sudo apt install cscope

Cscope: A Powerful code-searching tool

- in the source code directory,
  cscope -R
  will create cscope.out
- Once you have cscope.out, you can use cscope -d
  But if you have changed the source code, run cscope -R again
- **To exit, press Ctrl+D**