CS35L – Software Construction Lab

Winter 2016

TA – Sharath Gopal

CS35L – Course info

- Syllabus & detailed course information
  - Website (http://www.cs.ucla.edu/classes/winter16/cs35L/)
- Announcements – 'News' section of the website
- Piazza (https://piazza.com/) for class discussions
  - No sharing code or answers.
- Prerequisites – CS 31
- GNU/Linux distribution
  - Ubuntu 15.04 on CD.
  - You can use any other flavor of linux too
  - Installation Options
    - Boot from CD
    - Install on laptop on a separate partition (Take a backup!)
    - Install on virtual machine – VirtualBox
    - Windows users
      - Cygwin
- Get a SEASnet account asap
  - Add /usr/local/cs/bin to $PATH

CS35L – Course Info

- Assignments (10)
  - Lab exercises – Expected to be done in the lab
  - Homework
- All assignments to be done individually
- Submitted on CCLE
- Grading
  - Assignments – 50% (equally weighted)
  - Final exam – 50%
- Lateness penalty
  - 2^N % of the assignment's value for being 'N' days late
  - No assignments accepted after Friday of last week of instruction
- Assignment 1 and 10 are available
  - Assignment 1 – Due Friday (8th Jan)
  - Assignment 10 – Research and Development
    - Report and Class presentation
    - Resources at - http://cs.ucla.edu/classes/winter16/cs35L/comm.html

CS35L - Contact & Office Hours

- Sharath Gopal
  - sharath@cs.ucla.edu
- Office Hours
  - BH2432 – TBD

Introduction to Linux

Week 1 - Monday

What is an OS?

A Brief History of Operating Systems

- The Dark Ages
  - No OS until 1960s
  - Manually loaded programs
  - Reboot after each program
- Batch OS
  - Unified application development across systems
  - Output via printer, later via monitor
  - I/O via magnetic tape or disk
  - Written in assembler (e.g., OS/360)
  - Multiprocess

A Brief History of Operating Systems

- Timesharing OS
  - Multiuser
  - Multics (1964)
    - Segmented memory
    - Paged virtual memory
    - Applications written in many languages
    - Shared multiprocess memory
- Personal Computer
  - Single machine for single user
  - OS must manage screen and input devices
  - Window, Icon, Menu, Pointing Device (WIMP, e.g., MacOS, 1984)
- Cutting-Edge OS
  - High performance computer (HPC) clusters (e.g., BlueGene/L at LLNL rated at 280.6 teraFLOPS)
  - Cell phones, video
  - Video games
  - Browsers

Why Ubuntu?

- Multics (1964) → Unix (1970) → Minix (1987) → GNU/Linux (1991) → Ubuntu (2004)
- Debian based Linux
- Free software package via GNU
- Linux kernel (Unix-like OS)
- LiveCD Linux distribution
- Allows us to understand the workings of Unix
- Practice software construction via **command line interface** (CLI)

CLI vs. GUI

CLI	**GUI**
• Steep learning curve	• Intuitive
• Pure control (e.g., scripting)	• Limited Control
• Cumbersome multitasking	• Easy multitasking
• Speed: Hack away at keys	• Limited by pointing
• Convenient remote access	• Bulky remote access

Unix File System Layout

- Everything is a file (including devices)
- Tree structured hierarchy
- Lost? Man Pages
  - man: get manual or man pages
  - man ls : shows the man page for 'ls' command
  - /keyword : forward slash followed by keyword to search within a man page
  - q : quit the man page

The Basics: Moving Around

- pwd: print working directory
- cd: change working directory
- ~: home directory
- .: current directory
- /: root directory, or directory separator
- ..: parent directory

The Basics: Dealing with Files

- Environment variables
  - $PATH – List of directories to search for commands
  - $HOME – Home directory
- The basics continued…
  - printenv: prints all env variables
  - echo $PATH
  - echo $HOME
  - mv: move a file (no undos!)
  - cp: copy a file
  - rm: remove a file
  - mkdir: make a directory
  - rmdir: remove a directory
  - ls: list contents of a directory
    - -l: show long listing including permission info
    - -a: list all files including hidden ones
    - -s: show size of each file, in blocks
    - -h: human readable form

The Basics: File Name Matching

- ?: matches any single character in a filename
- *: matches one or more characters in a filename
- []*:* matches any one of the characters between the brackets. Use ‘-’ to separate a range of consecutive characters.

File/Directory Permissions

- User / Group / Others
  - User is the owner of the file
  - Group – csugrad
  - Other – others with accounts on the system
- rwx-rwx-rwx – 111 110 110
  - chmod 766 file.txt

The Basics: Command History

- <up arrow>: previous command
- <tab>: auto-complete
- !!: replace with previous command
  - ls
  - man !!

The Basics: Look These Up

Use man pages to see what these commands do.

- cat
- head
- tail
- du
- ps
- kill
- diff
- cmp
- wc
- sort

The Basics: Redirection

- > *file*: write stdout to a file
- >> *file*: append stdout to a file
- < *file*: use contents of a file as stdin

The Basics: Changing File Attributes

- ln: create a link
  - Hard links: points to physical data
  - Soft links aka symbolic links (-s): points to a file
- touch: update access & modification time to current time
- chmod
  - read (r), write (w), executable (x)
  - User, group, others

The Basics: find

- -type: type of a file (e.g,, directory, symbolic link)
- -perm: permission of a file
- -name: name of a file
- -prune: don’t descend into a directory
- -ls: list current file

Seasnet login options

- Remote Login via CLI
  - `ssh username@lnxsrv.seas.ucla.edu`
- Copy to/from seasnet server
  - `scp`
    - Usage similar to `cp`
      - `scp [source] [destination]`
    - Transferring files to remote host
      - `scp /home/username/doc.txt username@lnxsrv.seas.ucla.edu:/home/user/docs/`
    - Transferring files from remote host
      - `scp username@lnxsrv.seas.ucla.edu:/home/user/docs/foo.txt /home/username`
- Windows users
  - Cygwin
  - Putty
- Mac users
  - Terminal (you might have to install macports)
- Linux users
  - Terminal

Emacs

- Almost like a Windows text editor, but much more powerful
- Sometimes easier to use that vi

GNU Emacs Reference Card

(for version 20)

Starting Emacs

To enter GNU Emacs 20, just type its name: `emacs`

To read in a file to edit, see Files, below.

Leaving Emacs

suspend Emacs (or iconify it under X)	`C-z`
exit Emacs permanently	`C-x C-c`

Files

read a file into Emacs	`C-x C-f`
save a file back to disk	`C-x C-s`
save **all** files	`C-x s`
insert contents of another file into this buffer	`C-x i`
replace this file with the file you really want	`C-x C-v`
write buffer to a specified file	`C-x C-w`
version control checkin/checkout	`C-x C-q`

Getting Help

The help system is simple. Type `C-h` (or `F1`) and follow the directions. If you are a first-time user, type `C-h t` for a **tutorial**.

remove help window	`C-x 1`
scroll help window	`C-M-v`
apropos: show commands matching a string	`C-h a`
show the function a key runs	`C-h c`
describe a function	`C-h f`
get mode-specific information	`C-h m`

Error Recovery

abort partially typed or executing command	`C-g`
recover a file lost by a system crash	`M-x recover-file`
undo an unwanted change	`C-x u` or `C-_`
restore a buffer to its original contents	`M-x revert-buffer`
redraw garbaged screen	`C-l`

Incremental Search

search forward	`C-s`
search backward	`C-r`
regular expression search	`C-M-s`
reverse regular expression search	`C-M-r`
select previous search string	`M-p`
select next later search string	`M-n`
exit incremental search	`RET`
undo effect of last character	`DEL`
abort current search	`C-g`

Use `C-s` or `C-r` again to repeat the search in either direction. If Emacs is still searching, `C-g` cancels only the part not done.

Motion

entity to move over	**backward**	**forward**
character	`C-b`	`C-f`
word	`M-b`	`M-f`
line	`C-p`	`C-n`
go to line beginning (or end)	`C-a`	`C-e`
sentence	`M-a`	`M-e`
paragraph	`M-{`	`M-}`
page	`C-x [`	`C-x ]`
sexp	`C-M-b`	`C-M-f`
function	`C-M-a`	`C-M-e`
go to buffer beginning (or end)	`M-<`	`M->`

scroll to next screen	`C-v`
scroll to previous screen	`M-v`
scroll left	`C-x <`
scroll right	`C-x >`
scroll current line to center of screen	`C-u C-l`

Killing and Deleting

entity to kill	**backward**	**forward**
character (delete, not kill)	`DEL`	`C-d`
word	`M-DEL`	`M-d`
line (to end of)	`M-0 C-k`	`C-k`
sentence	`C-x DEL`	`M-k`
sexp	`M-- C-M-k`	`C-M-k`

kill region	`C-w`
copy region to kill ring	`M-w`
kill through next occurrence of *char*	`M-z` *char*
yank back last thing killed	`C-y`
replace last yank with previous kill	`M-y`

Marking

set mark here	`C-@` or `C-SPC`
exchange point and mark	`C-x C-x`
set mark *arg* **words** away	`M-@`
mark **paragraph**	`M-h`
mark **page**	`C-x C-p`
mark **sexp**	`C-M-@`
mark **function**	`C-M-h`
mark entire **buffer**	`C-x h`

Query Replace

interactively replace a text string	`M-%`
using regular expressions	`M-x query-replace-regexp`

Valid responses in query-replace mode are

replace this one, go on to next	`SPC`
replace this one, don't move	`,`
skip to next without replacing	`DEL`
replace all remaining matches	`!`
back up to the previous match	`^`
exit query-replace	`RET`
enter recursive edit (`C-M-c` to exit)	`C-r`

Multiple Windows

When two commands are shown, the second is for "other frame."

delete all other windows	`C-x 1`	
split window, above and below	`C-x 2`	`C-x 5 2`
delete this window	`C-x 0`	`C-x 5 0`
split window, side by side	`C-x 3`	
scroll other window	`C-M-v`	
switch cursor to another window	`C-x o`	`C-x 5 o`
select buffer in other window	`C-x 4 b`	`C-x 5 b`
display buffer in other window	`C-x 4 C-o`	`C-x 5 C-o`
find file in other window	`C-x 4 f`	`C-x 5 f`
find file read-only in other window	`C-x 4 r`	`C-x 5 r`
run Dired in other window	`C-x 4 d`	`C-x 5 d`
find tag in other window	`C-x 4 .`	`C-x 5 .`
grow window taller	`C-x ^`	
shrink window narrower	`C-x {`	
grow window wider	`C-x }`	

Formatting

indent current **line** (mode-dependent)	`TAB`
indent **region** (mode-dependent)	`C-M-\`
indent **sexp** (mode-dependent)	`C-M-q`
indent region rigidly *arg* columns	`C-x TAB`
insert newline after point	`C-o`
move rest of line vertically down	`C-M-o`
delete blank lines around point	`C-x C-o`
join line with previous (with arg, next)	`M-^`
delete all white space around point	`M-\`
put exactly one space at point	`M-SPC`
fill paragraph	`M-q`
set fill column	`C-x f`
set prefix each line starts with	`C-x .`
set face	`M-g`

Case Change

uppercase word	`M-u`
lowercase word	`M-l`
capitalize word	`M-c`
uppercase region	`C-x C-u`
lowercase region	`C-x C-l`

The Minibuffer

The following keys are defined in the minibuffer.

complete as much as possible	`TAB`
complete up to one word	`SPC`
complete and execute	`RET`
show possible completions	`?`
fetch previous minibuffer input	`M-p`
fetch later minibuffer input or default	`M-n`
regexp search backward through history	`M-r`
regexp search forward through history	`M-s`
abort command	`C-g`

Type `C-x ESC ESC` to edit and repeat the last command that used the minibuffer. Type `F10` to activate the menu bar using the minibuffer.

GNU Emacs Reference Card

Buffers

select another buffer	`C-x b`
list all buffers	`C-x C-b`
kill a buffer	`C-x k`

Transposing

transpose **characters**	`C-t`
transpose **words**	`M-t`
transpose **lines**	`C-x C-t`
transpose **sexps**	`C-M-t`

Spelling Check

check spelling of current word	`M-$`
check spelling of all words in region	`M-x ispell-region`
check spelling of entire buffer	`M-x ispell-buffer`

Tags

find a tag (a definition)	`M-.`
find next occurrence of tag	`C-u M-.`
specify a new tags file	`M-x visit-tags-table`
regexp search on all files in tags table	`M-x tags-search`
run query-replace on all the files	`M-x tags-query-replace`
continue last tags search or query-replace	`M-,`

Shells

execute a shell command	`M-!`
run a shell command on the region	`M-|`
filter region through a shell command	`C-u M-|`
start a shell in window `*shell*`	`M-x shell`

Rectangles

copy rectangle to register	`C-x r r`
kill rectangle	`C-x r k`
yank rectangle	`C-x r y`
open rectangle, shifting text right	`C-x r o`
blank out rectangle	`C-x r c`
prefix each line with a string	`C-x r t`

Abbrevs

add global abbrev	`C-x a g`
add mode-local abbrev	`C-x a l`
add global expansion for this abbrev	`C-x a i g`
add mode-local expansion for this abbrev	`C-x a i l`
explicitly expand abbrev	`C-x a e`
expand previous word dynamically	`M-/`

Regular Expressions

any single character except a newline	`.` (dot)
zero or more repeats	`*`
one or more repeats	`+`
zero or one repeat	`?`
quote regular expression special character *c*	`\`*c*
alternative ("or")	`|`
grouping	`\( ... \)`
same text as *n*th group	`\`*n*
at word break	`\b`
not at word break	`\B`

entity	**match start**	**match end**
line	`^`	`$`
word	`\<`	`\>`
buffer	`` \` ``	`\'`
class of characters	**match these**	**match others**
explicit set	`[ ... ]`	`[^ ... ]`
word-syntax character	`\w`	`\W`
character with syntax *c*	`\s`*c*	`\S`*c*

International Character Sets

specify principal language	`M-x set-language-environment`
show all input methods	`M-x list-input-methods`
enable or disable input method	`C-\`
set coding system for next command	`C-x RET c`
show all coding systems	`M-x list-coding-systems`
choose preferred coding system	`M-x prefer-coding-system`

Info

enter the Info documentation reader	`C-h i`
find specified function or variable in Info	`C-h C-i`

Moving within a node:

scroll forward	`SPC`
scroll reverse	`DEL`
beginning of node	`.` (dot)

Moving between nodes:

next node	`n`
previous node	`p`
move **up**	`u`
select menu item by name	`m`
select *n*th menu item by number (1–9)	*n*
follow cross reference (return with `l`)	`f`
return to last node you saw	`l`
return to directory node	`d`
go to any node by name	`g`

Other:

run Info **tutorial**	`h`
quit Info	`q`
search nodes for regexp	`M-s`

Registers

save region in register	`C-x r s`
insert register contents into buffer	`C-x r i`
save value of point in register	`C-x r SPC`
jump to point saved in register	`C-x r j`

Keyboard Macros

start defining a keyboard macro	`C-x (`
end keyboard macro definition	`C-x )`
execute last-defined keyboard macro	`C-x e`
append to last keyboard macro	`C-u C-x (`
name last keyboard macro	`M-x name-last-kbd-macro`
insert Lisp definition in buffer	`M-x insert-kbd-macro`

Commands Dealing with Emacs Lisp

eval **sexp** before point	`C-x C-e`
eval current **defun**	`C-M-x`
eval **region**	`M-x eval-region`
read and eval minibuffer	`M-:`
load from standard system directory	`M-x load-library`

Simple Customization

customize variables and faces	`M-x customize`

Making global key bindings in Emacs Lisp (examples):

```
(global-set-key "\C-cg" 'goto-line)
(global-set-key "\M-#" 'query-replace-regexp)
```

Writing Commands

```
(defun command-name (args)
  "documentation" (interactive "template")
  body)
```

An example:

```
(defun this-line-to-top-of-window (line)
  "Reposition line point is on to top of window.
With ARG, put point on line ARG."
  (interactive "P")
  (recenter (if (null line)
                0
              (prefix-numeric-value line))))
```

The `interactive` spec says how to read arguments interactively. Type `C-h f interactive` for more details.

vi

- Modes:
  - Normal: Enter commands
  - Insert: Insert text
  - Visual: Like normal, but you can highlight
  - Replace: Like insert, but you replace characters as you type
  - Recording: Record a sequence of key sequences

VI "Cheat" Sheet
ACNS Bulletin ED-03
February 1995

vi Editor "Cheat Sheet"

Invoking vi:	vi *filename*	
Format of vi commands:	*[count][command]*	(count repeats the effect of the command)

Command mode versus input mode

Vi starts in command mode. The positioning commands operate only while vi is in command mode. You switch vi to input mode by entering any one of several vi input commands. (See next section.) Once in input mode, any character you type is taken to be text and is added to the file. You cannot execute any commands until you exit input mode. To exit input mode, press the escape (**Esc**) key.

Input commands (end with Esc)

a	Append after cursor
i	Insert before cursor
o	Open line below
O	Open line above
:r *file*	Insert *file* after current line

Any of these commands leaves vi in input mode until you press **Esc**. Pressing the **RETURN** key will not take you out of input mode.

Change commands (Input mode)

cw	Change word (Esc)
cc	Change line (Esc) - blanks line
c$	Change to end of line
r*c*	Replace character with *c*
R	Replace (Esc) - typeover
s	Substitute (Esc) - 1 char with string
S	Substitute (Esc) - Rest of line with text
.	Repeat last change

Changes during insert mode

<ctrl>h	Back one character
<ctrl>w	Back one word
<ctrl>u	Back to beginning of insert

File management commands

:w *name*	Write edit buffer to file *name*
:wq	Write to file and quit
:q!	Quit without saving changes
ZZ	Same as :wq
:sh	Execute shell commands (<ctrl>d)

Window motions

<ctrl>d	Scroll down (half a screen)
<ctrl>u	Scroll up (half a screen)
<ctrl>f	Page forward
<ctrl>b	Page backward
/string	Search forward
?string	Search backward
<ctrl>l	Redraw screen
<ctrl>g	Display current line number and file information
n	Repeat search
N	Repeat search reverse
G	Go to last line
*n*G	Go to line *n*
:*n*	Go to line *n*
z<CR>	Reposition window: cursor at top
z.	Reposition window: cursor in middle
z-	Reposition window: cursor at bottom

Cursor motions

H	Upper left corner (home)
M	Middle line
L	Lower left corner
h	Back a character
j	Down a line
k	Up a line
^	Beginning of line
$	End of line
l	Forward a character
w	One word forward
b	Back one word
f*c*	Find *c*
;	Repeat find (find next *c*)

Deletion commands

dd or *n*dd	Delete *n* lines to general buffer
dw	Delete word to general buffer
d*n*w	Delete *n* words
d)	Delete to end of sentence
db	Delete previous word
D	Delete to end of line
x	Delete character

Recovering deletions

p	Put general buffer after cursor
P	Put general buffer before cursor

Undo commands

u	Undo last change
U	Undo all changes on line

Rearrangement commands

yy or Y	Yank (copy) line to general buffer
"*z*6yy	Yank 6 lines to buffer *z*
yw	Yank word to general buffer
"*a*9dd	Delete 9 lines to buffer *a*
"*A*9dd	Delete 9 lines; Append to buffer *a*
"*a*p	Put text from buffer *a* after cursor
p	Put general buffer after cursor
P	Put general buffer before cursor
J	Join lines

Parameters

:set list	Show invisible characters
:set nolist	Don't show invisible characters
:set number	Show line numbers
:set nonumber	Don't show line numbers
:set autoindent	Indent after carriage return
:set noautoindent	Turn off autoindent
:set showmatch	Show matching sets of parentheses as they are typed
:set noshowmatch	Turn off showmatch
:set showmode	Display mode on last line of screen
:set noshowmode	Turn off showmode
:set all	Show values of all possible parameters

Move text from file *old* to file *new*

vi *old*	
"*a*10yy	yank 10 lines to buffer *a*
:w	write work buffer
:e *new*	edit new file
"*a*p	put text from *a* after cursor
:30,60w *new*	Write lines 30 to 60 in file *new*

Regular expressions (search strings)

^	Matches beginning of line
$	Matches end of line
.	Matches any single character
*	Matches any previous character
.*	Matches any character

Search and replace commands

Syntax:

```
:[address]s/old_text/new_text/
```

Address components:

.	Current line
n	Line number n
.+m	Current line plus m lines
$	Last line
/string/	A line that contains "string"
%	Entire file
[addr1],[addr2]	Specifies a range

Examples:

The following example replaces only the first occurrence of Banana with Kumquat in each of 11 lines starting with the current line (.) and continuing for the 10 that follow (.+10).

```
:.,.+10s/Banana/Kumquat
```

The following example replaces every occurrence (caused by the `g` at the end of the command) of `apple` with `pear`.

```
:%s/apple/pear/g
```

The following example removes the last character from every line in the file. Use it if every line in the file ends with `^M` as the result of a file transfer. Execute it when the cursor is on the first line of the file.

```
:%s/.$//
```