CS 35L - Winter 2016

TA: Tomer Weiss
Jan-07-2016

goo.gl/Otw6Xz

slides link

CS 35L - Course information

- Assignment 1 is be available
  - http://web.cs.ucla.edu/classes/winter16/cs35L/assign/assign1.html
- Deadline: Jan-08-2016

PTE

- Will pass sign-in sheet
- Attend each class and sign your name
- PTE given during second week of classes

Seasnet important notice

- Login and do your Homework on the following servers:
  - ssh username@lnxsrv07.seas.ucla.edu
  - ssh username@lnxsrv09.seas.ucla.edu
- Seasnet is upgrading other seasnet servers
- We are going to test your assignment solutions on these servers

Seasnet

Secure Remote Login File Transfer

For secure remote login and
file transfer, use ssh and sftp (instead of telnet and ftp).

To run graphical
application on a remote unix server, see X11 Forwarding.

Windows Clients

- PuTTY SSH
  - How to install
  - How to use
- WinSCP freeware SFTP and SCP client for Windows
- X11 Forwarding
- Xming X Server for Windows

Unix Clients

- Example: how to use ssh
- Example: how to use sftp

Macintosh Clients

- Note that Mac OS X includes OpenSSH by default.
- OpenSSH Mac OS clients

www.seasnet.ucla.edu/secure-remote-login-file-transfer/

Introduction to Linux

Week 1
Part 2

Emac - shortcuts (only for Mac)

- Home : fn+control+Left Arrow (Works exactly similar to Windows Home key)
- End : fn+control+Right Arrow
- Page Up : fn+Up Arrow
- Page Down : fn+Down Arrow

Learning to use Emacs - Pointers

- Navigating with file
- Searching file
  - C-s, C-r
- Erasing a line
  - C-k to erase from current cursor to end of line

Visiting Emacs scratch buffer

- GNU Emacs default bindings:
  - C-x b scratch RET
- Or alternatively
  - M-x switch-to-buffer scratch

Learning to use Emacs - Pointers

- Compiling ***C*** code with emacs
  - M-x compile
- Copying emacs buffer into file
- Running Lisp code
  - M-x emacs-lisp-mode
  - C-x C-e : Evaluate expression up to point
  - http://www.emacswiki.org/emacs/EvaluatingExpressions

Learning to use Emacs - Pointers

- Lisp eval helpful pointers
  - http://www.emacswiki.org/emacs/EvaluatingExpressions
  - http://www.gnu.org/software/emacs/manual/html_node/emacs/Lisp-Eval.html

Lab

http://web.cs.ucla.edu/classes/winter16/cs35L/assign/assign1.html

The Basics: Look These Up

- 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
  - Also used to create a file
- 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(s)
- Reminder: man find

Seasnet login option

- 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 /home/username/
      - Windows users
        - » Cygwin
        - » Putty
      - Mac users
        - » Terminal (might need to install mac-ports)
      - Linux users
        - » Terminal

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
"*a*6yy	Yank 6 lines to buffer *a*
yw	Yank word to general buffer
"*a*9dd	Delete 9 lines to buffer *a*
"A9dd	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/.$//
```