
CS 35L- Software Construction Lab 3

Fall 18

TA: Guangyu Zhou

Course Information

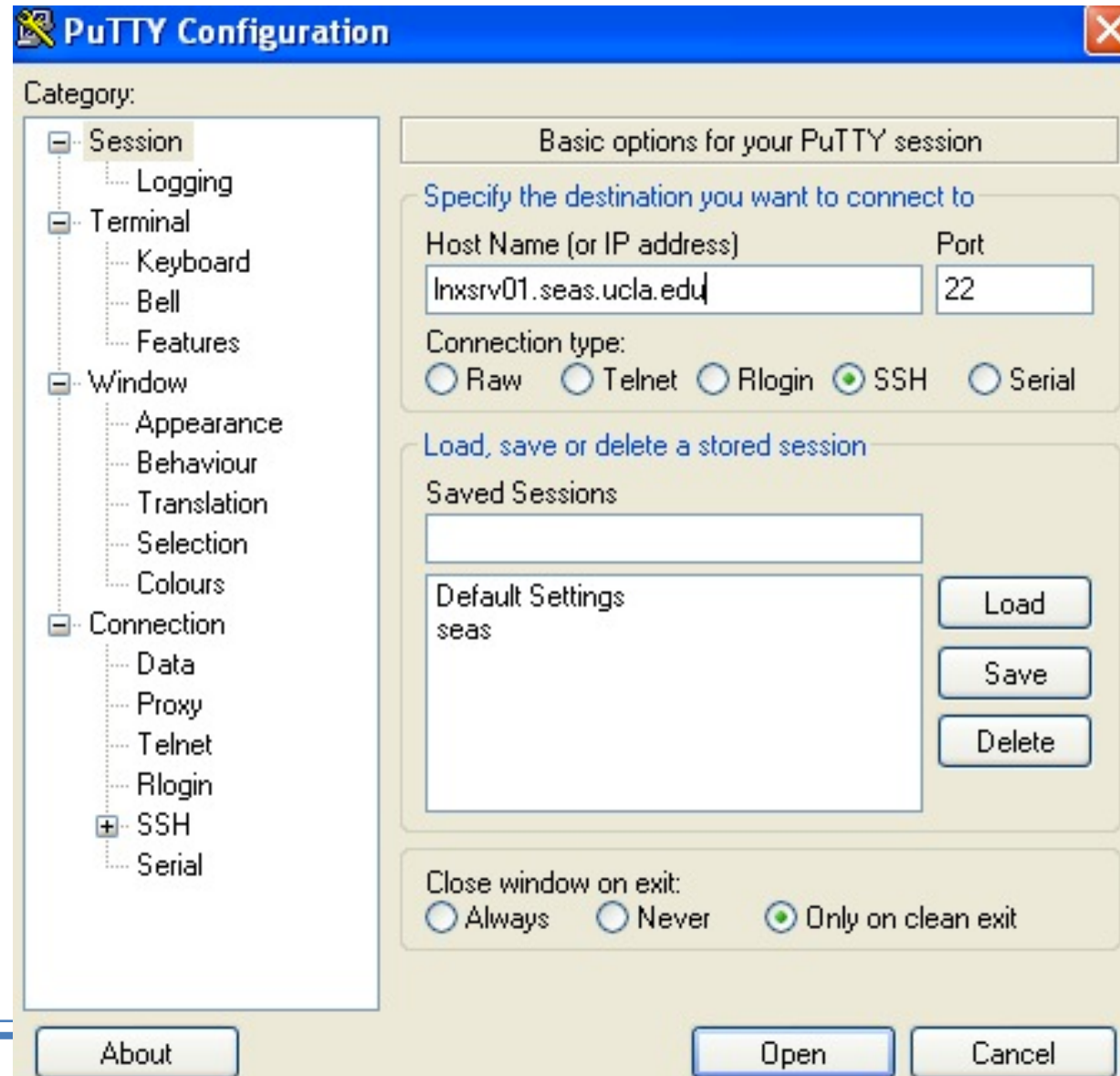
- Contact information
 - Email: guangyuzhou@g.ucla.edu
 - **Office hour:** Wed 11am-1pm, BH 3256S-E
 - Course Website: <http://web.cs.ucla.edu/classes/fall18/cs35L/index.html>
 - Join Piazza(<https://piazza.com>): for class discussions
 - Search for CS35L, fall 2018
 - **Use Piazza to discuss about assignments or course related questions!**
-

Course Information

- Prepend /usr/local/cs/bin to \$PATH (for all future assignment)
 - Login and do your homework on the following server
 - Inxsrv06, Inxsrv07, Inxsrv07, or Inxsrv10,
 - ssh username@Inxsrv*.seas.ucla.edu
 - Submit your assignment to CCLE by deadline.
 - For people who haven't enrolled, please send the assignment to the grading TA by e-mail with you UID, and Name.
-

Connecting to SEAS from Windows

PuTTY



Connecting to SEAS from OS X or Linux

- Terminal
 - \$ ssh username@lnxsrv.seas.ucla.edu
 - Username = your SEAS user name

Some tips for preappend PATH

To preappend PATH, run the following command:

```
export PATH=/usr/local/cs/bin:$PATH  
echo $PATH
```

And it should have "/usr/local/cs/bin" appended to your path.

If you use the above approach, every time you open a new terminal or logged into the Inxsr0x, you need to execute the command starting with export again.

To avoid the trouble, you can modify the .bashrc and .bash_profile file under your home directory.

1. Open the .bashrc file under your home directory: `emacs ~/.bashrc`
2. Append "export PATH=/usr/local/cs/bin:\$PATH" at the end of .bashrc file
3. Save and exit from emacs.
4. Repeat 1,2,3 on the .bash_profile file.
5. Logged out from the Inxsr and then logged back in.
6. Run `echo $PATH` to verify that /usr/local/cs/bin is the first field of the \$PATH and separated from the latter field with a column.

Note: If you do not have .bash_rc and the .bash_profile under your home directory, the above command will create one for you. So that should be fine.

.bash_profile vs .bashrc

- **.bash_profile** is executed for login shells, while **.bashrc** is executed for interactive non-login shells.
 - When you login (type username and password) via console, either sitting at the machine, or remotely via ssh: **.bash_profile** is executed to configure your shell before the initial command prompt.
 - But, if you've already logged into your machine and open a new terminal window (xterm) then **.bashrc** is executed before the window command prompt. **.bashrc** is also run when you start a new bash instance by typing `/bin/bash` in a terminal.
 - Tips:
 - Better to want to do PATH adjustments in **.bash_profile** instead of **.bashrc**, since these changes are typically not [idempotent](#).
 - **export PATH="\$PATH:/some/addition"** If you put that in **.bashrc** instead, every time you launched an interactive sub-shell, `:/some/addition` would get tacked on to the end of the PATH again, creating extra work for the shell when you mistype a command.
-

Introduction to Linux

Week 1

What is Linux

- Operating system
 - Created by Linus and a group of people (online)
 - Unix-like open source software
 - Free to contribute, free to use
 - Four Components (Linux distribution)
 - Linux kernel
 - GNU utilities
 - Graphical desktop environment
 - Application software
-

Component I: Linux kernel

- Functionalities
 - Software program management
 - Hardware management
 - Filesystem management
 - System memory management
 - “top” command

```
Tasks: 375 total, 1 running, 337 sleeping, 37 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 65795584 total, 60219548 free, 845064 used, 4730972 buff/cache
KiB Swap: 20479996 total, 20479996 free, 0 used. 64253224 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10889	guangyuz	20	0	166472	2660	1680	R	0.3	0.0	0:00.11	top
1	root	20	0	192180	5228	2600	S	0.0	0.0	1:50.82	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.08	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.05	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0.0	0.0	0:04.06	kworker/u64:0
8	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/0
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
10	root	20	0	0	0	0	S	0.0	0.0	0:41.30	rcu_sched
11	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	lru-add-drain
12	root	rt	0	0	0	0	S	0.0	0.0	0:01.25	watchdog/0
13	root	rt	0	0	0	0	S	0.0	0.0	0:01.22	watchdog/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.49	migration/1
15	root	20	0	0	0	0	S	0.0	0.0	0:00.03	ksoftirqd/1
17	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/1:0H
19	root	rt	0	0	0	0	S	0.0	0.0	0:01.12	watchdog/2
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration/2
21	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/2
23	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/2:0H
24	root	rt	0	0	0	0	S	0.0	0.0	0:01.09	watchdog/3
25	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	migration/3
26	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/3
28	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/3:0H
29	root	rt	0	0	0	0	S	0.0	0.0	0:01.12	watchdog/4
30	root	rt	0	0	0	0	S	0.0	0.0	0:00.05	migration/4
31	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/4
33	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/4:0H
34	root	rt	0	0	0	0	S	0.0	0.0	0:01.10	watchdog/5
35	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	migration/5
36	root	20	0	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/5
38	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/5:0H
39	root	rt	0	0	0	0	S	0.0	0.0	0:01.05	watchdog/6
40	root	rt	0	0	0	0	S	0.0	0.0	0:00.06	migration/6

Component II: GNU utilities

- System utilities to run on Linux kernel
- Contains coreutils package
 - Handling files (touch, rm, mkdir, ls ...)
 - Manipulating text (tr, sed ...)
- Shell is a special interactive utility (CLI)
 - Bash is the default shell in Linux
 - Bourne shell (sh)
 - C shell (csh)
 - TC shell (tcsh)
 - Korn shell (ksh)
 - Bourne Again shell (bash)
 - Type 'echo \$0' to see which shell you are using!
 - You can open shells recursively

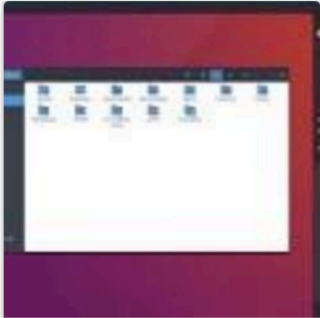





-
- Reference: <https://www.gnu.org/software/coreutils/coreutils.html>

Component III & IV: Graphical desktop environment & Application software

linux Graphical desktop environment

[All](#) [Images](#) [Videos](#) [News](#) [Maps](#) [More](#) [Settings](#) [Tools](#)

Desktop environment Software > linux

					
Budgie GNU Lesser...	LXQt GNU Gener...	GNOME GNU Gener...	Xfce BSD licenses	LXDE Freeware	Cinnamon GNU Gener...

Review of basic commands + Demo

- man
 - pwd
 - cd:
 - ~
 - .
 - /
 - ..
-

Review of basic commands + Demo

- The basics continued...
 - mv:
 - cp:
 - rm: **rm removes files blindly, with no concept of 'trash'.**
 - rm -r:
 - mkdir:
 - rmdir:
 - ls:
 - -d: list only directories
 - -a: list all files including hidden ones
 - -l: show long listing including permission info
 - -s: show size of each file, in blocks
-

Review of basic commands + Demo

- **Touch [Option] [File(s)] :**
 - If file exists:
 - update access & modification time to current time
 - `touch -t 201101311759.30 filename`
 - Change filename's access & modification time to (year 2011 January day 31 time 17:59:30)
 - If file not exists:
 - An empty file is created
 - **ln:** create a link
 - Hard links: point to physical data
 - Soft links aka symbolic links (-s): point to a file
-

Review: Soft link vs. hard link

```
$ ls -li
total 20
9962464 -rw-r--r-- 2 guru users 8 Mar 9 file1
9962464 -rw-r--r-- 2 guru users 8 Mar 9 file2
9962471 lrwxrwxrwx 1 guru users 5 Mar 9 file3 -> file1
```

Hard link:

```
File1-----\_____|inode|-----|welcome|
                |_____|
File2-----/_____|9962464|
```

Soft link:

```
File1-----|_____|-----|welcome|
                |_____|
                9962464

File3-----|_____|-----|File1|
                |_____|
                9962471
```


- Create two files

\$ touch blah1

\$ touch blah2

-
- Fill contents into the files and print them

\$ echo "Cat" > blah1

\$ echo "Dog" > blah2

\$cat blah1; cat blah2

Cat

Dog

- Create links:

- ***\$ ln blah1 blah1-hard***

\$ ln -s blah2 blah2-soft

\$ ls -li

blah1 blah1-hard

blah2 blah2-soft -> blah2

- Change the original file

\$ mv blah1 blah1-new

\$ cat blah1-hard

Cat

\$ mv blah2 blah2-new

\$ cat blah2-soft

cat: blah2-soft: No such file or directory

wh... Commands

```
→ ~ man -k "wh" | grep "^wh"
```

what (1)	- show what versions of object modules were used to construct a file
whatis (1)	- search the whatis database for complete words
whereis (1)	- locate programs
which (1)	- locate a program file in the user's path
while (ntcl)	- Execute script repeatedly as long as a condition is met
who (1)	- display who is logged in
whoami (1)	- display effective user id
whois (1)	- Internet domain name and network number directory service

Linux File Permissions

```
shum@sol:~$ ls -l
total 20
drwx----- 2 shum staff 4096 Jan 16 22:04 Mail
drwx----- 3 shum staff 4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum staff 4096 Jan 13 16:42 public
drwxr-xr-x  2 shum staff 4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum staff 628 Jan 15 20:04 verse
```

Annotations for the `ls -l` output:

- File type:** Indicated by the first character of the permission string (yellow arrow pointing to `d` for directory and `-` for regular file).
- User (owner) permissions:** Indicated by the first three characters after the file type (green arrow pointing to `rw` for the owner of the `verse` file).
- Group permissions:** Indicated by the next three characters (cyan arrow pointing to `r--` for the group of the `verse` file).
- Other (everyone) permissions:** Indicated by the last three characters (red arrow pointing to `r--` for others of the `verse` file).
- Number of hard links:** The number between the permission string and the user name (white arrow pointing to `1` for the `verse` file).
- User (owner) name:** The user name (white arrow pointing to `shum` for the `verse` file).
- Group name:** The group name (white arrow pointing to `staff` for the `verse` file).
- Size:** The file size in bytes (white arrow pointing to `628` for the `verse` file).
- Date/time last modified:** The date and time (white arrow pointing to `Jan 15 20:04` for the `verse` file).
- Filename:** The file name (white arrow pointing to `verse` for the `verse` file).

Legend for permissions:

rwX

- r**: readable
- w**: writeable
- X**: executable

The Basics: chmod (numeric)

#	Permission
7	full
6	read and write
5	read and execute
4	read only
3	write and execute
2	write only
1	execute only
0	none

- Usage
 - `chmod ["references"]["operator"]["modes"] "file1" ...`
 - Example: **chmod** ug+rw mydir, **chmod** a-w myfile,
 - Example: **chmod** ug=rx mydir, **chmod** 664 myfile
-

Review: Processes

- Everything is either a process or a file in Linux:
 - Process
 - An instance of a computer program in execution
 - ps
 - List processes that are currently running
 - kill
 - Terminate a certain process
 - Usage
 - kill PID
-

diff: command

- A file comparison utility that outputs the differences between two files.
 - Shows the changes between one version of a file and a former version of the same file
 - Usage
 - `diff original_file new_file`
 - `diff -u original_file new_file` (unified format)
-

- **file1.txt:**

- I need to go to the store.

- I need to buy some apples.

- When I get home, I'll wash the dog.

- **file2.txt:**

- I need to go to the store.

- I need to buy some apples.

- Oh yeah, I also need to buy grated cheese.**

- When I get home, I'll wash the dog.

- `diff file1.txt file2.txt`

- 2a3

- > Oh yeah, I also need to buy grated cheese.

-
- **file1.txt:**
 - I need to go to the store.
 - I need to buy some apples.
 - When I get home, I'll wash the dog.
 - I promise.**
 - **file2.txt:**
 - I need to go to the store.
 - I need to buy some apples.
 - When I get home, I'll wash the dog.
 - `diff file1.txt file2.txt`
 - 4d3
 - < I promise.
-

- **file1.txt:**

I need to buy apples.

I need to run the laundry.

I need to wash the dog.

I need to get the car detailed.

- **file2.txt:**

I need to buy apples.

I need to do the laundry.

I need to wash the car.

I need to get the dog detailed.

```
diff file1.txt file2.txt
```

```
2,4c2,4
```

```
< I need to run the laundry.
```

```
< I need to wash the dog.
```

```
< I need to get the car detailed.
```

```
---
```

```
> I need to do the laundry.
```

```
> I need to wash the car.
```

```
> I need to get the dog detailed.
```

- **file1.txt:**

apples

oranges

kiwis

carrots

- **file2.txt:**

apples

kiwis

carrots

grapefruits

- `diff -u file1.txt file2.txt`

--- file1.txt 2014-08-21 17:58:29.764656635 -0400

+++ file2.txt 2014-08-21 17:58:50.768989841 -0400

@@ -1,4 +1,4 @@

apples

-oranges

kiwis

carrots

+grapefruits

wget

- A computer program that retrieves content from web servers
 - Usage
 - `wget <URL>`
-

Introduction to Linux text editors: 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
 - Supplement material: vi editor cheat sheet
-

Emacs

- Almost like a Windows text editor, but much more powerful
 - Different from vi
 - Only one mode
 - Command: use control buttons:
 - Ctrl(**C**- for short)
 - Alt or Option (also known as Meta, **M**- for short)
 - Supplement material: GNU Emacs Reference Card
 - <https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf>
 - Vim vs Emacs:
 - <https://stackoverflow.com/questions/1430164/differences-between-emacs-and-vim>
-

GNU Emacs Reference Card

(for version 26)

Starting Emacs

To enter GNU Emacs 26, just type its name: **emacs**

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
toggle read-only status of buffer	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
describe the function a key runs	C-h k
describe a function	C-h f
get mode-specific information	C-h m

Error Recovery

abort partially typed or executing command	C-g
recover files lost by a system crash	M-x recover-session
undo an unwanted change	C-x u , C-_ 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 matched.

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, top, bottom		C-l
goto line		M-g g
goto char		M-g c
back to indentation		M-m

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-O 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 <i>char</i>		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 <i>arg</i> 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 or y
replace this one, don't move	,
skip to next without replacing	DEL or n
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 a similar command for a frame instead of a window.

delete all other windows	C-x 1	C-x 5 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 <i>arg</i> columns	C-x TAB
indent for comment	M-;
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 <i>arg</i> , next)	M-^
delete all white space around point	M-\
put exactly one space at point	M-SPC
fill paragraph	M-q
set fill column to <i>arg</i>	C-x f
set prefix each line starts with	C-x .
set face	M-o

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
regex search backward through history	M-r
regex 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 menu bar items on text terminals.

Emacs

“The customizable, extensible, self documenting, real-time display editor”

- Customizable (no programming)
 - Users can customize font, colors, etc. in ~/.emacs
 - Extensible (programming required)
 - Run Lisp scripts to define new commands (dired)
 - Self-documenting
 - C-h r (manual) and C-h t (tutorial)
 - Real-time
 - Edits are displayed on screen as they occur
-

Learning to use Emacs - Pointers

- Navigating with file
 - Move up/down/left/right: C-p, C-n, C-b, C-f (arrow keys also work)
 - Move to the beginning/end of a line: C-a, C-e
 - Move to the first/last line of the text: M-<, M-> (use shift for < and >)
 - Search and replace file
 - C-s: search forward
 - C-r: search backward
 - M-%: replace (usage: M-% [source] Enter [dest])
 - Erasing a line
 - C-k: erase from current cursor to end of line
-

Learning to use Emacs

- Copy and paste in a file
 - Begin: C-@ or C-Space (press Ctrl+Shift+2)
 - Use the <up> and <down> buttons to select the contents
 - End: C-w (cut), M-w(copy), C-y (paste)
 - Undo command: C-u

Note: the “paste” option in Emacs is named as **yanking**

Extended Commands

- Use C-x plus other combined button
 - New file: C-x C-f
 - Quit Emacs: C-x C-c
 - If a file is modified, it will ask you whether to save the file and whether to leave now.
(input y, yes)
-

Extended Command

- Visiting Emacs scratch buffer:
 - Copy(save) current buffer to file
C-x C-s
 - List all the buffers
C-x C-b
 - Save current buffer with a specified file name
C-x C-w [filename]
 - Add(read) a new file to buffer
C-x C-f
-

Learning to use Emacs - Compiling

- Visit/switch to *scratch* buffer

C-x b

- Compiling **C** code with emacs

- M-x compile

- Running Lisp code

- M-x emacs-lisp-mode

- C-x C-e : Evaluate expression up to point

<http://www.emacswiki.org/emacs/EvaluatingExpressions>

```
~/demos/emacs_usage > ls  
~/demos/emacs_usage > █
```

Lab1 hints for first 10 questions:

- | | |
|--|---|
| 1. What shell command uses the <code>man</code> program to print all the commands that have a specific word in their man page (or at least the description part of the man page)? (hint: <code>man man</code>) | 1. <code>man man</code> |
| 2. Where are the <code>cp</code> and <code>sh</code> programs located in the file system? List any shell commands you used to answer this question. | 2. <code>which</code> |
| 3. What executable programs have names that are just one character long, and what do they do? List any shell commands you used to answer this question. | 3. <code>find</code> |
| 4. When you execute the command named by the symbolic link <code>/usr/bin/emacs</code> , which file actually is executed? List any shell commands you used to answer this question. | 4. <code>readlink</code> |
| 5. What is the version number of the <code>/usr/bin/emacs</code> program? of the plain <code>emacs</code> program? Why are they different programs? | 5. <code>emacs --version</code> |
| 6. The <code>chmod</code> program changes permissions on a file. What does the symbolic mode <code>g+s,o-x</code> mean, in terms of permissions? | 6. <code>man chmod</code> |
| 7. Use the <code>find</code> command to find all directories modified in the last 30 days that are located under (or are the same as) the directory <code>/usr/local/cs</code> . List any shell commands you used to answer this question. | 7. <code>find</code> |
| 8. Of the files in the same directory as <code>find</code> , how many of them are symbolic links? List any shell commands you used to answer this question. | 8. <code>whereis</code> , <code>man find</code> |
| 9. What is the oldest regular file in the <code>/usr/lib64</code> directory? Use the last-modified time to determine age. Specify the name of the file without the <code>/usr/lib64/</code> prefix. Consider files whose names start with <code>"."</code> . List any shell commands you used to answer this question. | 9. <code>find</code> , <code>sort</code> |
| 10. Where does the <code>locale</code> command get its data from? List any shell commands you used to answer this question. | 10. <code>localedef</code> |
-

Homework1: Format of submission

- For exercises (key1.txt)
 - Record keystroke of each exercise separately
 - Don't forget commands to enter/leave emacs
 - The keystrokes for different exercise should be recorded separately, each keystroke for one line

Exercise 1.1

```
1. e m a c s SP assign1.html Enter
2. C-s T
3. C-b
```

.

.

.

```
11. C-x C-c
```

Exercise 1.2

```
1. e m a c s SP assign2.html Enter
```

.

.

.

```
5. ..... Backspace Backspace
```

Some General Tips

- Just use simple commands, this is a warm-up task. Take it easy.
 - If you are not sure about complex commands, just use the combination of simple ones
e.g. <left> <left> <left> = C-a
 - If you don't know the exact answers, just write down what you thought and what you have tried
-

Tips for Emacs Exercises

- Exercise 1.1
 - Basic Navigation Commands
M-b M-< C-e C-a
 - Search Commands
C-s
 - Exercise 1.2
 - Find some thing and delete
First use search command, then use the command to delete a line
 - Jump to given line
M-x goto-line [number]
-

Tips for Emacs Exercises

- Exercise 1.3

- Search and replace command

C-s M-%

If you are not familiar with M-%, just use delete plus insert

- Exercise 1.4

- Copy and paste (cut)
 - Search and replace
-

Tips for Emacs Exercises

- Exercise 1.5
 - M-x compile
 - Delete the “make -k” command, type in “gcc -o [bin file name] [source file name]”
 - Buffer commands: Save current buffer with a specified file name
 - Exercise 1.6
 - Evaluate expressions
 - C-x C-e
 - Be aware of the maximum number in 32-bit integer
-

Week1 Check List

GUI & CLI basics

Unix file system layout

Unix permission

Basic commands

Documentations and man pages

Emacs basics

Supplement Resources

- [emacs commands.pdf](#)
- [unix ref.pdf](#)
- [vi_cheat sheet.pdf](#)