CS35L Lab4

Spring 2017

Administration

- Instructor's office hour:

  Mondays 14:00–15:00 and Thursdays 09:30–10:30

  @BH 4532J

- Assignment 1 is due Saturday 4/8 11:55pm on CCLE
- http://web.cs.ucla.edu/classes/spring17/cs35L/assign.html
- Two files: ans1.txt, key1.txt -- manually graded

Work using your own computer

- Mac user

- Download ssh client

- ssh accountname@ugrad.seas.ucla.edu

- Windows user

- Download PUTTY

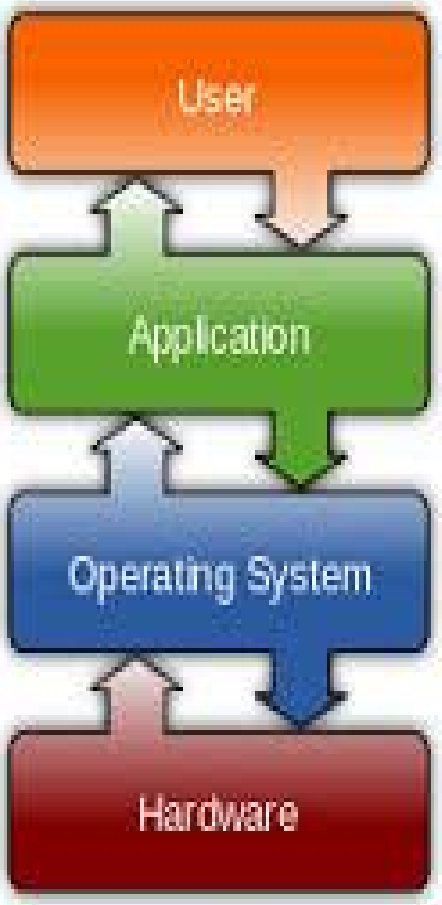
- Do similar things as above

- Make sure your code works on the server before submitting!

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



Linux Kernel

- Four main functionalities
  - System memory management
  - Software program management
  - Hardware management
  - Filesystem management

GNU utilities

- System utilities to run on linux kernel
- Contains coreutils package
  - Handling files
  - Manipulating text
  - Managing text
- Shell is a special interactive utility (CLI)
  - Bash is the default shell in Linux

Graphical desktop environment

- Two common graphical environment
  - KDE
  - GNOME desktop

Linux File System Layout

- Stores files in a single virtual directory
  - Does not use drive letters in pathnames
- Tree structured hierarchy
- Everything is a file (including devices)
- Each user has a separate home directory
  - A new shell starts from home directory

Common Linux Directory Names

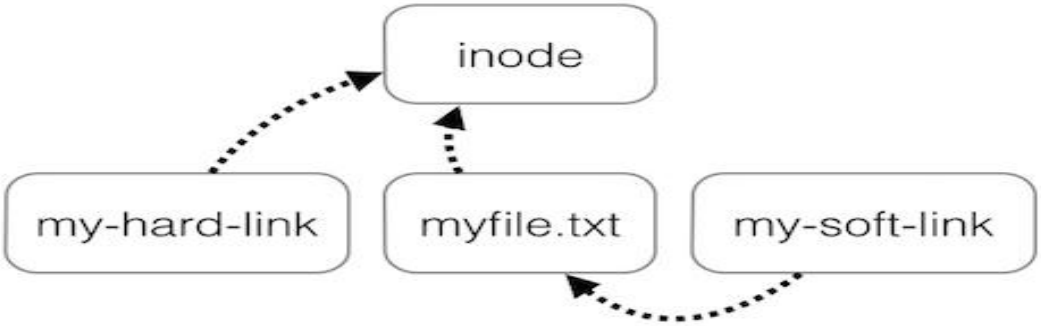
Directory	Usage
/	The root of the virtual directory. Normally, no files are placed here.
/bin	The binary directory, where many GNU user-level utilities are stored.
/boot	The boot directory, where boot files are stored.
/dev	The device directory, where Linux creates device nodes.
/etc	The system configuration files directory.
/home	The home directory, where Linux creates user directories.
/lib	The library directory, where system and application library files are stored.
/media	The media directory, a common place for mount points used for removable media.
/mnt	The mount directory, another common place for mount points used for removable media.
/opt	The optional directory, often used to store optional software packages.
/root	The root home directory.
/sbin	The system binary directory, where many GNU admin-level utilities are stored.
/tmp	The temporary directory, where temporary work files can be created and destroyed.
/usr	The user-installed software directory.
/var	The variable directory, for files that change frequently, such as log files.

Path in linux file system

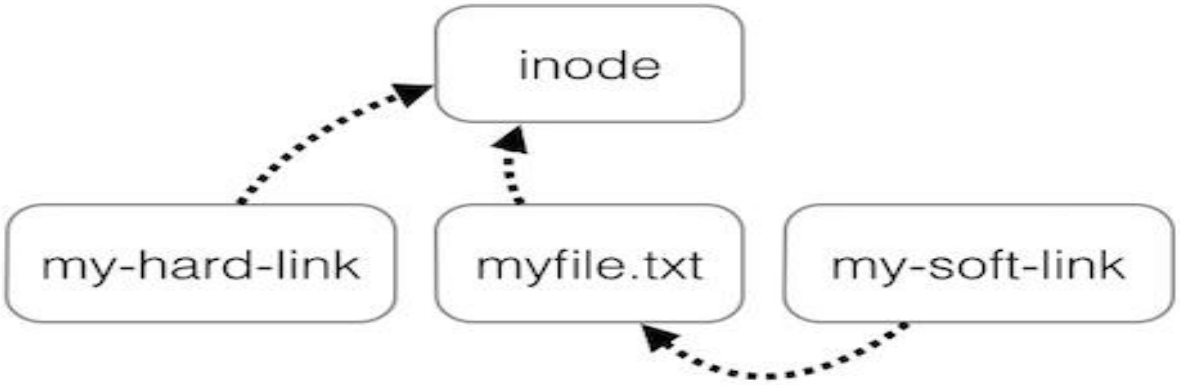
- Absolute path
  - Path from root directory
- Relative path
  - Path relative to current directory

File

- touch filename
  - Create an empty file
  - Update the modification and access time of file
- File is represented as inode (index node)
  - Each inode has an inode number
  - ls -i
- File != File name
  - Many file names can refer to the same copy of data

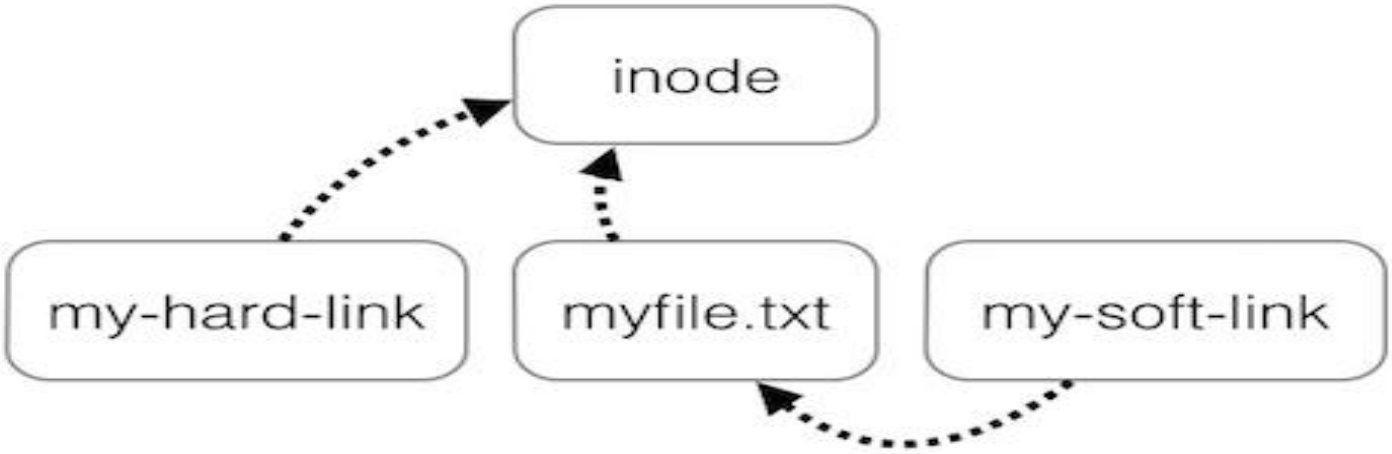


- Two types of links
  - Hard links: points to physical data (inode)
  - Soft/symbolic link (-s): points to a file
- ln *file filename*
  - create a hard link by default
  - option -s creates soft link
- readlink *filename*
  - Resolve symbolic link



Question:

- Does cp command create links or copy data?
- What is the potential problem of running rm command in the case?



File Attributes

- File attribute: many (list some?)
- stat: display file status

```
  File: 'test.txt'
  Size: 22              Blocks: 0          IO Block: 65536  regular file
Device: 1fh/31d Inode: 4910057     Links: 1
Access: (0644/-rw-r--r--)  Uid: (10422/ cs35lt4)   Gid: (  300/ taaccts)
Access: 2017-04-05 23:00:11.047000000 -0700
Modify: 2017-04-05 23:00:11.047996000 -0700
Change: 2017-04-05 23:00:11.048995000 -0700
 Birth: -
```

- file: check file type
- Text files
- Executable files
- Data files

File Attributes

Permission: limit of access for different users

- 3 users : user (u), group (g), others (o)
- 3 permission: read (r, 4), write (w, 2),executable (x, 1)

- chmod
  - change file modes
  - symbolic mode: chmod u+x *filename*
  - numerical mode: chmod 751 *filename*

Search files in a directory

- find
- Walk down a file hierarchy
- Options
  - type: type of a file (directory, symbolic link)
  - perm: permission of a file
  - name: name of a file
  - prune: don’t descend into a directory
  - ls: list current file

File Name Matching in option name

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

The Basics: History

- <up arrow>: previous command
- <tab>: auto-complete
- !!: replace with previous command
- ![*str*]: refer to previous command with str
- ^[*str*]: replace with command referred to as str

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/.$//
```

GNU Emacs

- A free, portable, extensible text editor
- More powerful than a text editor
  - A programming environment
  - Can compile, evaluate, debug etc.
- Only one mode
- Two important keys: M(Alt or Meta), C(Ctrl)
- Emacs cheatsheet:

https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf

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 nth 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	\sc	\Sc

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 nth 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.

For copies of the GNU Emacs manual, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111 1307 USA

Emacs basics

- C-t : tutorial
- C-x C-c: exit
- Emacs filename
- C-g: cancel partially typed or accidental command
- C-x u: undo last change

Emacs on files

- Navigating with file
  - Move up/down/left/right: C-p, C-n, C-b, C-f (or arrow keys)
  - Move to the beginning/end of a line: C-a, C-e
  - Move to the first/last line of the text: M-<, M->
- 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
- Copy and paste in a file
  - Begin: C-@ (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

Extended commands

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

Compile with Emacs

- Visit *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

Format of submission

- For lab questions(ans1.txt)
  - Answer 15 questions using natural language
  - List all the commands used to solve the problem
  - Give some explanations about your choice of commands
  - No need for keystrokes in this file
  - Will be graded manually

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

```
Exersise 1.1
1. e m a c s SP assign1.html Enter
2. C-s T
3. C-b
.
.
.
11. C-x C-c

Exersise 1.2
1. e m a c s SP assign2.html Enter
.
.
.
5. ..... Backspace Backspace
```

Some 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

Submit your homework

- Use scp to copy files from server to your local machine
  - Google how to use
- Submit your two files in ccle
- Lab file must be named as ans1.txt

  Homework file must be named as key1.txt

Week1 Check List

- Multiuser and multiprocess OS
- GUI basics
- CLI basics
- Unix file system layout
- Unix permission
- Basic commands
- Documentations and man pages
- Emacs basics