Linux Quick-start Guide

Yonsei University, ELC lab, in particular Sung-ho Kim.

Contents

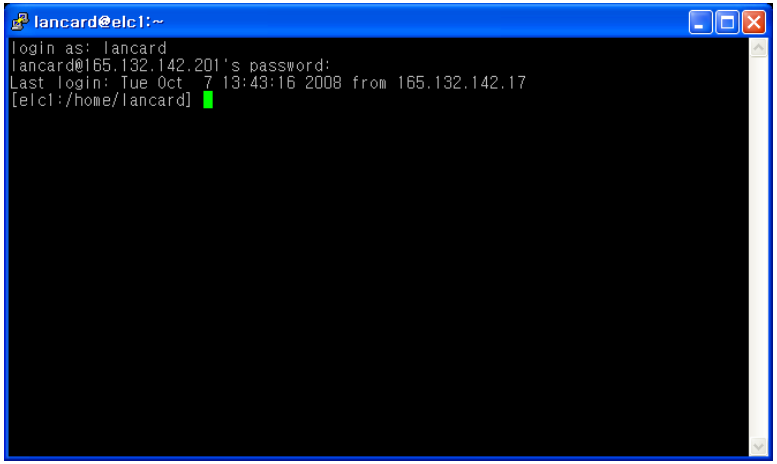
- What is Linux?
- Logging in, File Transfers
- Structure of Directories
- Commands
- Using Editors

What is Linux

- A fully-networked 32/64-Bit Unix Operating System
- Multi-user, Multitasking, Multiprocessor
- Comes with the X Window GUI
- Can coexists with other Operating Systems on your computer
  - ``dual boot'' setups
- Runs on multiple platforms
- It’s free!
- Includes the Source Code (``open-source'')

Logging In

- See the lecture slides on Assignment 1 for logging in on our server using putty or ssh



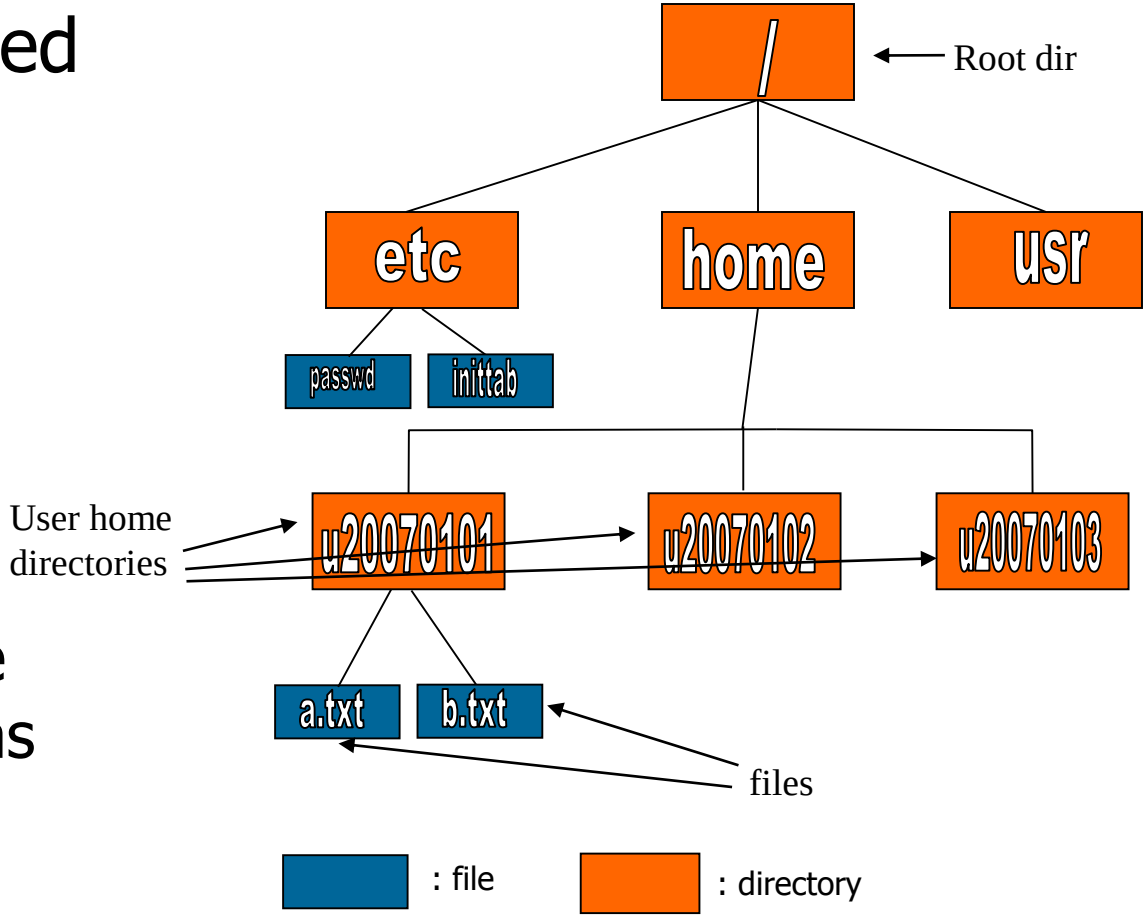

```
[my: ~]# ssh lancard@elc1.cs.yonsei.ac.kr
lancard@elc1.cs.yonsei.ac.kr's password:
Last login: Wed Oct  8 12:11:15 2008 from 165.132.142.17
[elc1:/home/lancard]$
```

File Transfers

- Used to transfer files between computers
  - Example: from your laptop to the elc1 server
- File transfer protocols
  - Ex) ftp, sftp, scp, ...
- Because of security reasons, only scp is widely used nowadays (also on our server).
  - To copy a file foo.txt from your computer to the server:
    - scp foo.txt <username>@elc1.cs.yonsei.ac.kr:
  - To copy a file foo.txt from the server to your computer:
    - scp <username>@elc1.cs.yonsei.ac.kr:foo.txt .
    - (Don't forget to include the period at the end, it denotes the current working directory on your computer!)

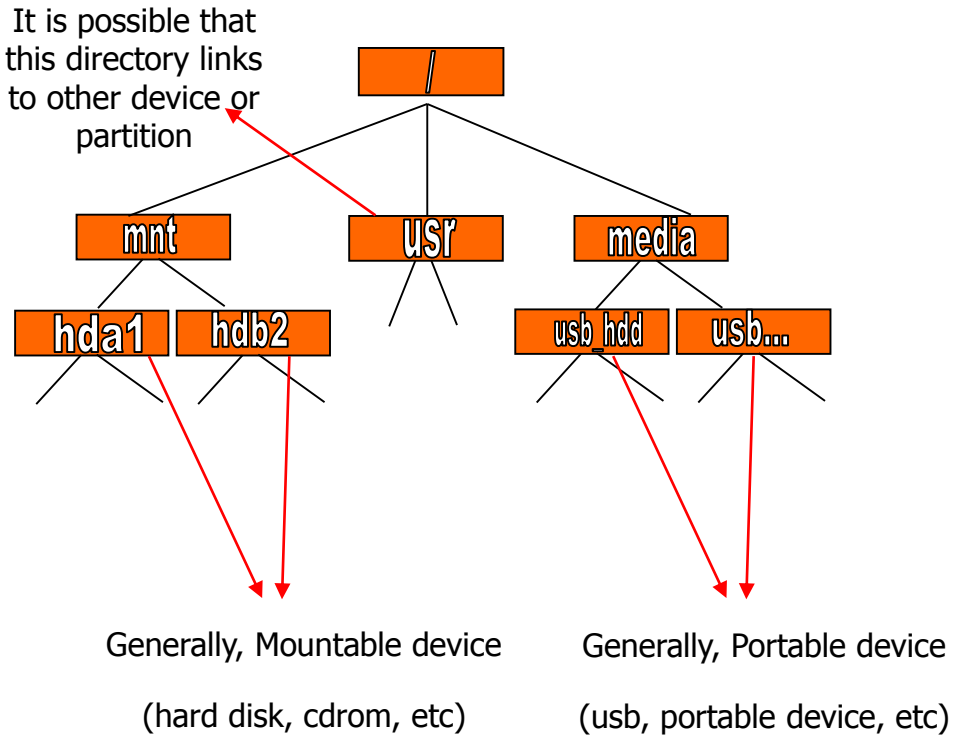
Linux File System Basics

- Linux files are stored in a single rooted, hierarchical file system
  - Files are stored in directories (folders)
  - Directories may be nested as deeply as needed

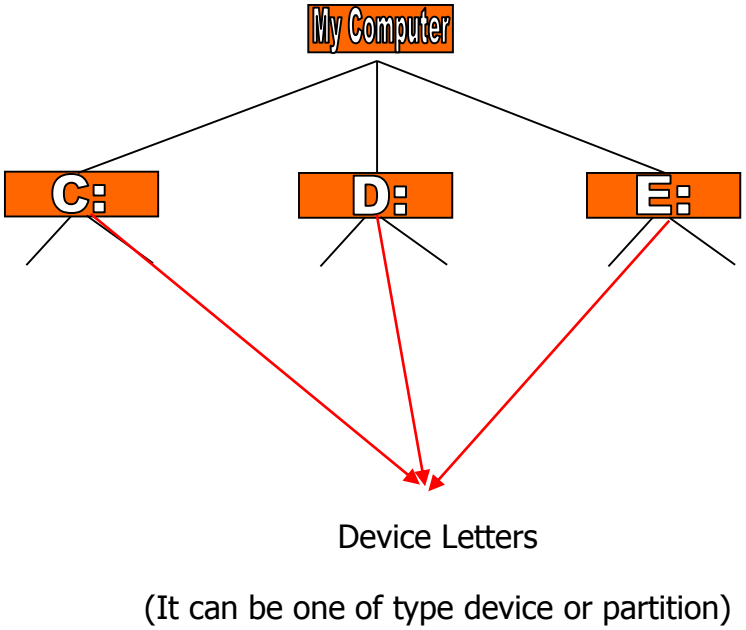


Windows and Linux FSs

- Linux can mount (links to any type of storage) to any directories.
  (but generally, LINUX systems uses '/mnt' and '/media' directories to mount)



<Linux System>



<Windows System>

Some Special File Names

- Some file names are special:
  - `/` The root directory
  - `.` The current directory
  - `..` The parent (previous) directory
  - `~` My home directory (such as /home/ccugrad/u20070101)
- Examples:
  - `./a` same as `a`
  - `../jane/x` go up one level then look in directory `jane` for `x`

Special Files

- `/home` - all users' home directories are stored here (however, in our case, we use '/home/ccugrad/')
- `/bin`, `/usr/bin` – directories that contain system commands
- `/sbin`, `/usr/sbin` – further system commands
- `/etc` - all sorts of configuration files
- `/var` - logs, spool directories etc.
- `/dev` - device files
- `/proc` – in-memory files provided by the kernel for configuration and monitoring

Linux Command Basics

- To execute a command, type its name and arguments at the command line

```
ls -l /etc
```

Command name

Options (flags)

Arguments

Basic Commands - pwd

- Print Working (Current) Directory

```
[student_sample@elc1 ~]# pwd

/homes/pp/student_sample
```

Basic Commands - ls

- List directory contents

```
[student_sample@elc1 ~]# ls
kkk.txt
[student_sample@elc1 ~]# ls -l
total 9432
-rw-r--r-- 1 root root      10 Sep 22 02:47 kkk.txt
[student_sample@elc1 ~]#
```

permission | Owner | size | File name

Basic Commands - cp

- Copy file(s)

```
[student_sample@elc1 ~]# cp kkk.txt kkk2.txt
[student_sample@elc1 ~]# cp /homes/assignments/assignment2/sample.ppm .
```

Copy file “kkk.txt” to “kkk2.txt”

Copy file “sample.ppm” from directory ‘/homes/assignments/assignment2 to the current working directory (“.”) .

Basic Commands - mv

- Move file(s)

```
[student_sample@elc1 ~]# mv kkk.txt kkk2.txt
[student_sample@elc1 ~]# mv ./sample.ppm /tmp
```

Move kkk.txt to kkk2.txt

Move ./sample.ppm to the /tmp directory.

Basic Commands - rm

- Remove file(s)

```
[student_sample@elc1 ~]# rm kkk.txt
```

Remove kkk.txt

Basic Commands - mkdir

- Make Directory

```
[student_sample@elc1 ~]# mkdir sample
[student_sample@elc1 ~]# cd sample
[student_sample@elc1 ~]# pwd
/homes/pp/student_sample/sample
```

Basic Commands - rmdir

- Remove Directory (empty)

```
[student_sample@elc1 ~]# rmdir sample
```

Basic Commands - cat

- Concatenate files and print on the standard output

  (general case, print contents of file)

```
[student_sample@elc1 ~]# cat a.txt
1 2 3 4 5
```

Processes - PID

- PID
  - A process ID is a unique identifier assigned to a process while it runs
  - Each time you run a program, Linux assignes a different PID (it takes a long time for a PID to be reused by the system)
  - You can use the PID to track the status of a process with the `ps` command or `top`, or to end a process with the `kill` command

Basic Commands - ps

- Report a snapshot of the current processes. (Process Snapshot)

```
[elc1:/homes/pp] ps
  PID TTY          TIME CMD
16806 pts/3    00:00:00 bash
21362 pts/3    00:00:00 ps
[elc1:/homes/pp] ps xl
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
5   501 16762 16756  15   0 104296  2428 -      S    ?          0:00 sshd: lancard@pts/2
0   501 16763 16762  15   0  84128  2684 wait   Ss   pts/2      0:00 -bash
0   501 16801 16763  15   0  71736 14340 -      R+   pts/2      0:00 ssh lancard@elc1.cs.yonsei.ac.kr
5   501 16805 16802  15   0 104296  2384 -      S    ?          0:00 sshd: lancard@pts/3
                                          ait   Ss   pts/3      0:00 -bash
                                                S    ?          0:00 sshd: lancard@notty
                                                Ss   ?          0:00 /usr/libexec/openssh/sftp-server
                                                S    ?          0:00 sshd: lancard@notty
                                                Ss   ?          0:00 /usr/libexec/openssh/sftp-server
0   501 21365 16806  17   0   8344   724 -      R+   pts/3      0:00 ps xl
[elc1:/homes/pp]
```

x option: show all my processes (not only current login state)

l option: print detail information.

Basic Commands - kill

- kill process

```
[student_sample@elc ~]# ps
UID        PID  PPID  C STIME TTY          TIME CMD
root      6715  6692  2 14:34 ttyp0    00:00:00 sleep 10h
root      6716  6692  0 14:34 ttyp0    00:00:00 ps -ef
[student_sample@elc ~]# kill 6715
[1]+  Terminated              sleep 10h
[student_sample@elc ~]#
```

Basic Commands - top

- display Linux tasks
  (continuously update per 1 sec)

```
top - 17:07:36 up 9 days, 21:50,  4 users,  load average: 2.00, 1.77, 1.01
Tasks: 182 total,   3 running, 178 sleeping,   1 stopped,   0 zombie
Cpu(s): 25.0%us,  0.1%sy,  0.0%ni, 74.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  20544600k total,  7245284k used, 13299316k free,   336876k buffers
Swap: 65537156k total,        0k used, 65537156k free,  6254268k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0 10324  676  572 S    0  0.0   0:02.55 init
    2 root      RT  -5     0    0    0 S    0  0.0   0:00.02 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
```

More Commands

- `find` - Searches a given file hierarchy specified by path, finding files that match the criteria given by expression

```
find ./ -name "*.c"
```

In this example, we look for files with an extension ".c" (that is, C source files).

More Commands

- `diff` – find differences between two files

```
[elc1:/home/lancard] cat a.txt
1 2 3
[elc1:/home/lancard] cat b.txt
1 2 3
[elc1:/home/lancard] cat c.txt
1 2 3 4
[elc1:/home/lancard] diff a.txt b.txt
[elc1:/home/lancard] diff a.txt c.txt
1c1
< 1 2 3
---
> 1 2 3 4
[elc1:/home/lancard]
```

More Commands

- grep - Searches files for one or more pattern arguments.

```
[elc1:/home/lancard] cat a.txt
1 2 3
[elc1:/home/lancard] cat b.txt
1 2 3
[elc1:/home/lancard] cat c.txt
1 2 3 4
[elc1:/home/lancard] grep 2 *.txt
a.txt:1 2 3
b.txt:1 2 3
c.txt:1 2 3 4
[elc1:/home/lancard] grep 4 *.txt
c.txt:1 2 3 4
[elc1:/home/lancard]
```

More Commands

- `tar` - manipulates archives

```
[student_sample@elc ~]# tar -cvf sample.tar sample
sample/
sample/a.txt
sample/b.txt
Sample/c.txt
Sample/test/a.txt
```

< archiving 'sample' directory to sample.tar >

```
[student_sample@elc ~]# tar -xvf sample.tar
sample/
sample/a.txt
sample/b.txt
Sample/c.txt
Sample/test/a.txt
```

< extracting 'sample' directory from sample.tar >

File Permissions

- Every file
  - Is owned by someone
  - Belongs to a group
  - Has certain access permissions for owner, group, and others

File Permissions

- Linux provides three kinds of permissions:
  - Read - users with read permission may read the file or list the directory
  - Write - users with write permission may write to the file or add new files to the directory
  - Execute - users with execute permission may execute the file or lookup a specific file within a directory

File Permissions

- The long version of a file listing (`ls -l`) will display the file permissions:

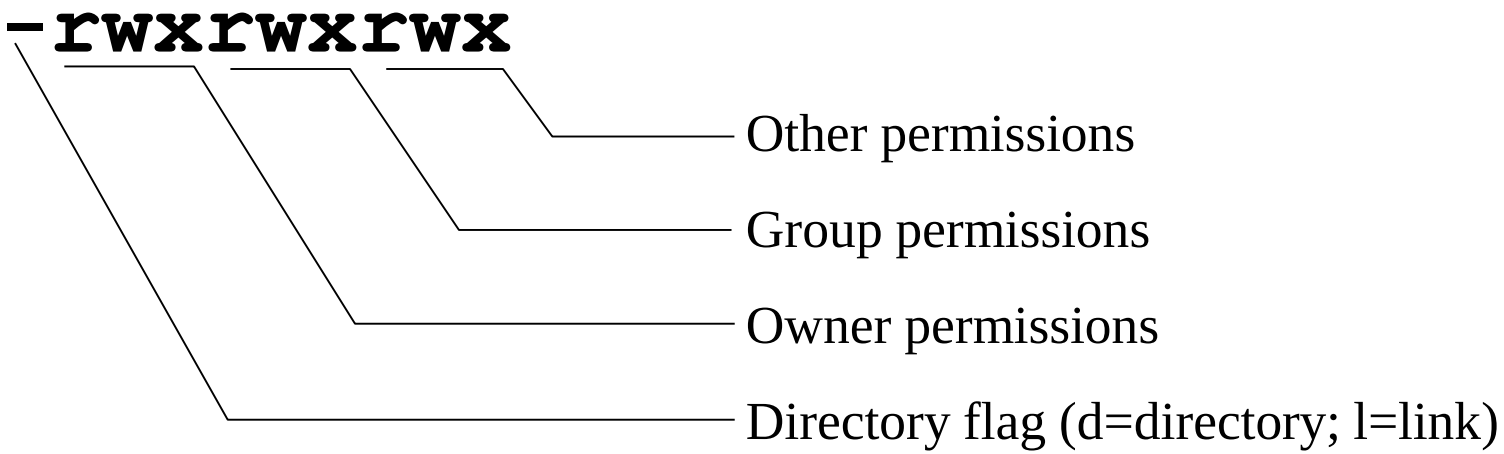
```
-rwxrwxr-x   1 rvdheij  rvdheij      5224 Dec 30 03:22 hello
-rw-rw-r--   1 rvdheij  rvdheij       221 Dec 30 03:59 hello.c
-rw-rw-r--   1 rvdheij  rvdheij      1514 Dec 30 03:59 hello.s
drwxrwxr-x   7 rvdheij  rvdheij      1024 Dec 31 14:52 posixuft
```

Permissions

Owner

Group

Interpreting File Permissions



Changing File Permissions

- Use the `chmod` command to change file permissions
  - The permissions are encoded as an octal number

```
chmod 755 file   # Owner=rwx Group=r-x Other=r-x
chmod 500 file   # Owner=r-x Group=--- Other=---
chmod 644 file   # Owner=rw- Group=r-- Other=r--

chmod +x  file   # Add execute permission to file for all
chmod o-r file   # Remove read permission for others
chmod a+w file   # Add write permission for everyone
```

Editors

- People are passionate about their editor
- Several choices are available:
  - `vi` Standard UNIX editor
  - `emacs` Extensible, Customizable Self-Documenting Editor
  - `nano` Simple text editor

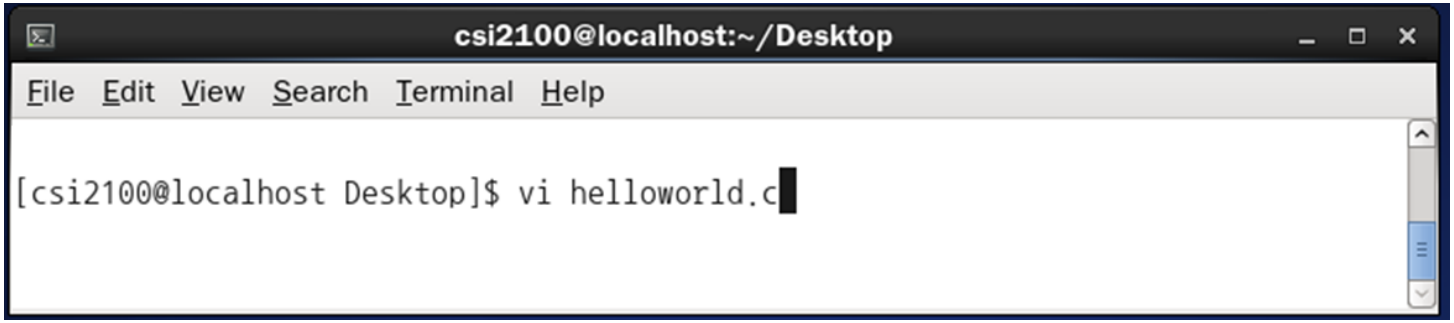
(You're strongly recommended to use vi, as it is very common and efficient, once you know it well.)

Vi editor

- Sample edit session
- VI in detail

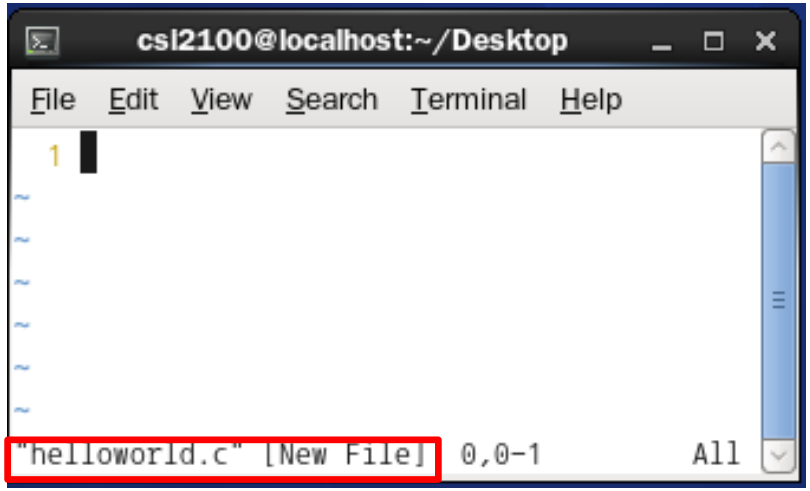
VI sample edit session: (1.) open a file

- Decide on the name of your sourcefile, e.g., helloworld.c
- Type '**vi helloworld.c**' in the terminal. This will
  - open 'helloworld.c' if the file exists
  - create and open 'helloworld.c' if the file does not exist

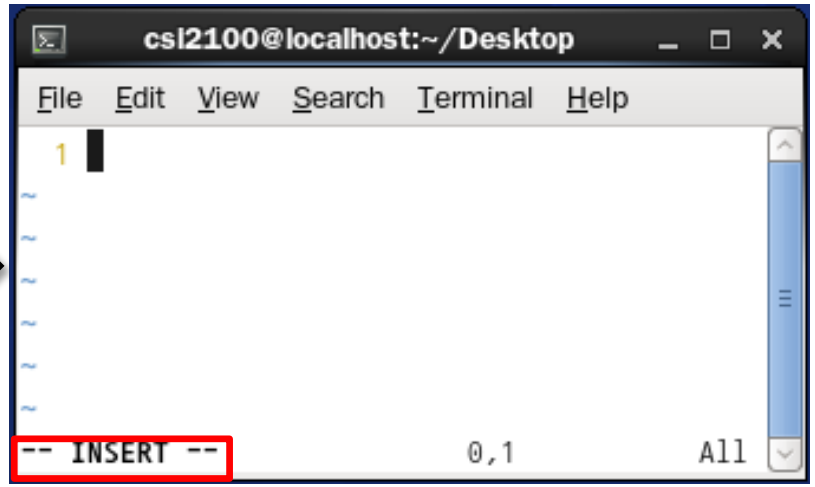


VI sample session: (2.) Switching VI modes

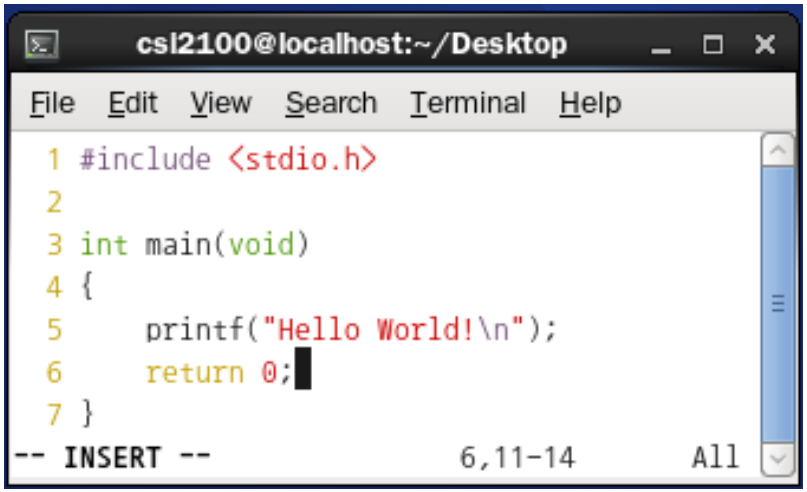
- Switch vi to **insert mode** by pressing '**i**':



press 'i'

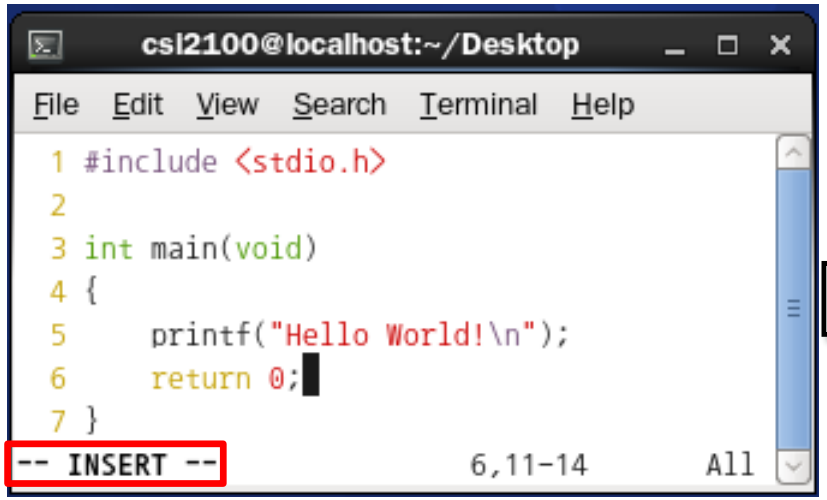


- Now, you can type text (editing the file) ☺

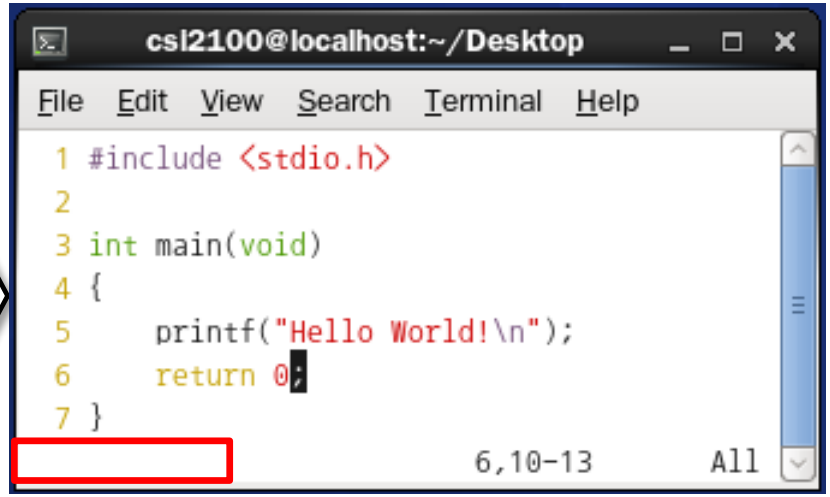


VI sample session: (3) Save and Quit

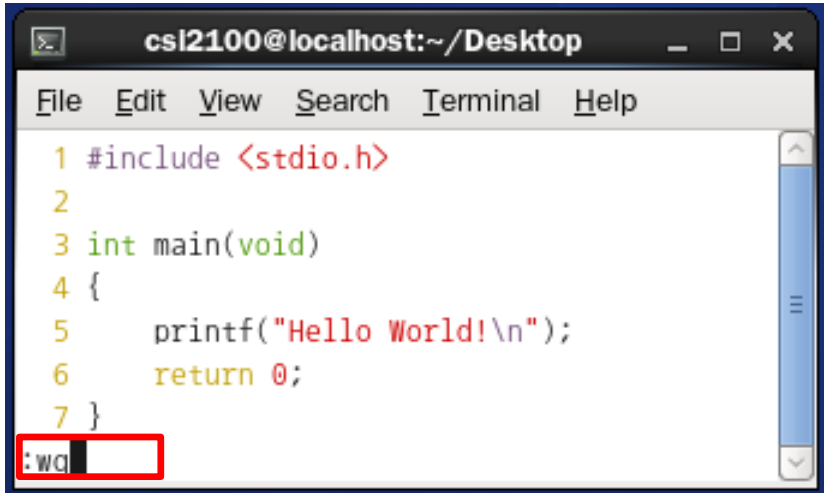
- Switch back to vi to **command mode** by pressing the '**esc**' key:



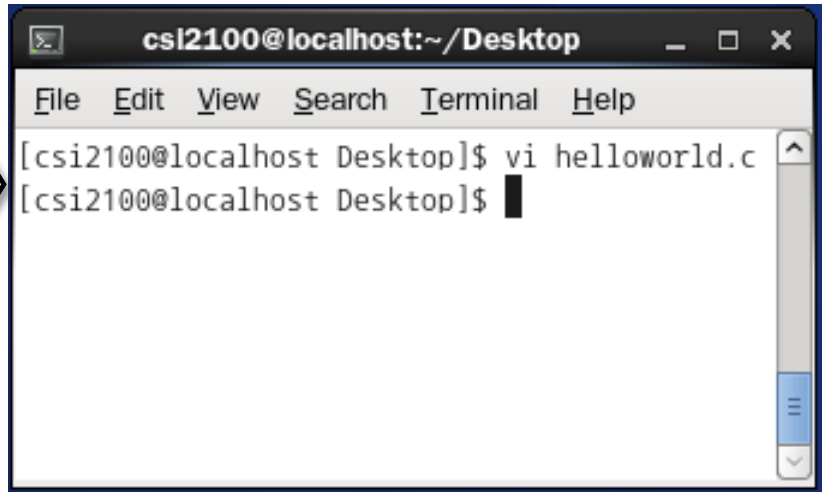
press 'esc'



- Type '**:wq**' to write and quit (close); SHORTCUT: hold **shift** and press **zz**



press 'enter'



Vi in detail

- Starting the Vi editor
  - Command to start Vi

    [elc1: ~]$ vi
  - Opening an existing file or new file

    [elc1: ~]$ vi {filename}

Vi editor

- Exiting the Vi editor
  - Quit the editor

    :q
  - Quit the editor without saving the changes to the file.

    :q!
  - Quit the editor with saving the changes

    :wq

Vi editor

- Modes of Operation
  - **Command Mode**

    Allows the entry of commands to manipulate text

    Commands are usually one or two characters long
  - **Insert Mode**

    Puts anything you type on the keyboard into the current file

Vi editor

- Vi starts in command mode by default
- Most commonly used commands to get into **insert mode** are *'a' and 'I'*
  - *press 'a' or 'i' key to enter insert mode.*
- Once in insert mode, you get out of it by hitting the Esc key

Vi Editor

- Some simple Vi Commands
  - a

    Enter insert mode; the characters typed will be inserted after the current cursor position. If a count is specified, then the inserted text will be repeated that many times
  - i

    Enter insert mode, the characters typed will be inserted before the current cursor position. If a count is specified, the inserted text will be repeated that many times
  - x

    Delete character under the cursor. Count specifies how many characters to delete

Vi Editor

- Some simple Vi Commands
  - r

    replace one character under the cursor. Specify count to replace that many characters.
  - R

    Starting from the current cursor position, replace the characters with the one typed on the keyboard
  - u

    undo the last change to the file

Vi Editor

- Some simple Vi Commands
  - h

    Move the cursor to the left one character position
  - l

    Move the cursor to the right one character position
  - j

    Move the cursor down one line
  - k

    Move the cursor up one line

Vi Editor

- Cutting text
  - d^

    Deletes from current cursor position to the beginning of the line
  - d$

    Deletes from current cursor position to the end of the line
  - Dw

    Deletes from current cursor position to the end of the word
  - dd

    Deletes one line from current cursor position. Specify count to delete many lines.

Vi Editor

- Yanking (Copying) and Pasting
  - yl

    yank a single character. Specify count to yank more characters
  - yw

    yank a single word. Specify count to yank more words
  - yy

    yank a single line. Specify count to yank more lines
  - p

    paste the text that was either deleted or yanked previously

Vi Editor

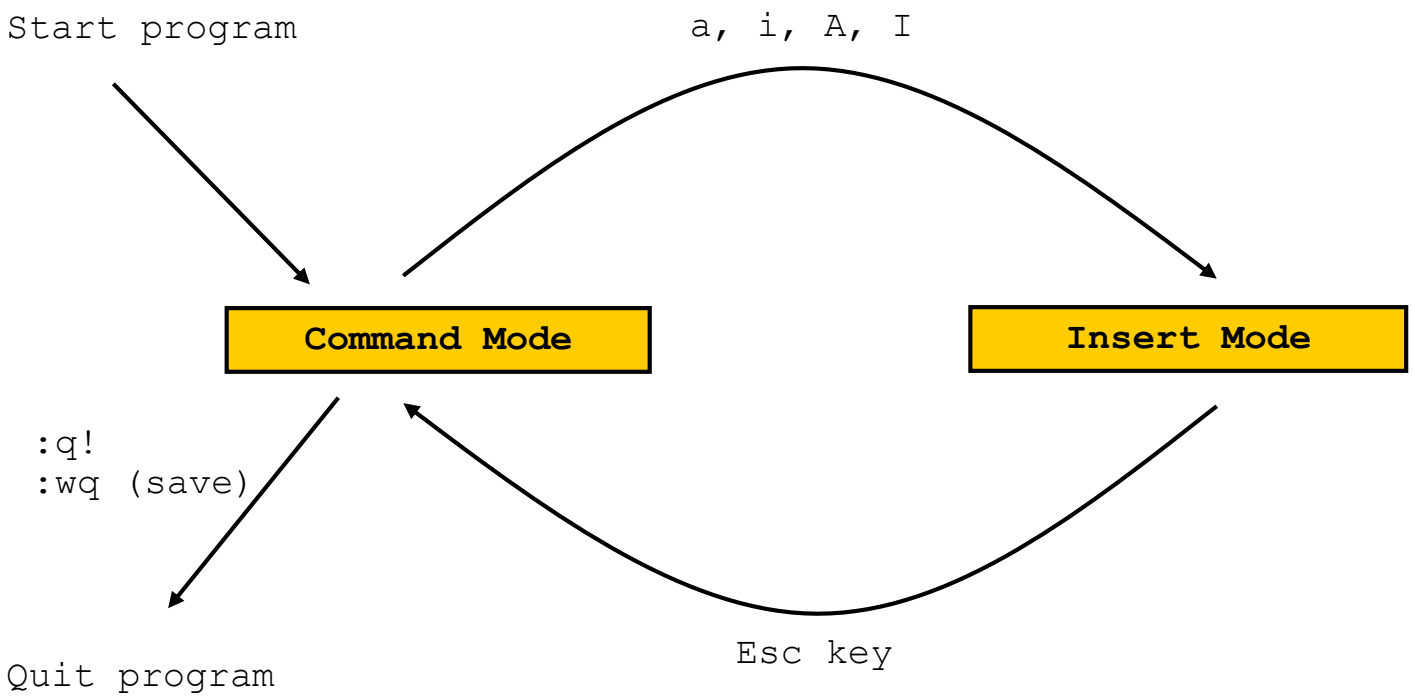
- To go to a specific line in the file

  :*linenumber*

- *String Search*
  - */[pattern] : search forward for the pattern*
  - *?[pattern] : search backward for the pattern*

Vi Editor

- Basic Diagram



Vi Editor

- Sample Typing

(You can see '—INSERT—' when you are in insert mode)

```
Hello this is sample
~
~
~
~
~
~
~
~
:wq
```

<Command mode>

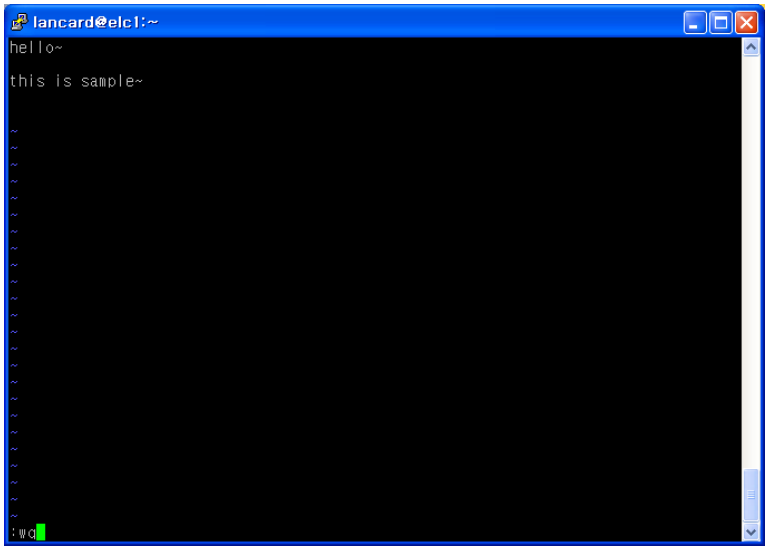
```
Hello this is sample
~
~
~
~
~
~
~
~
--INSERT--
```

<Insert mode>

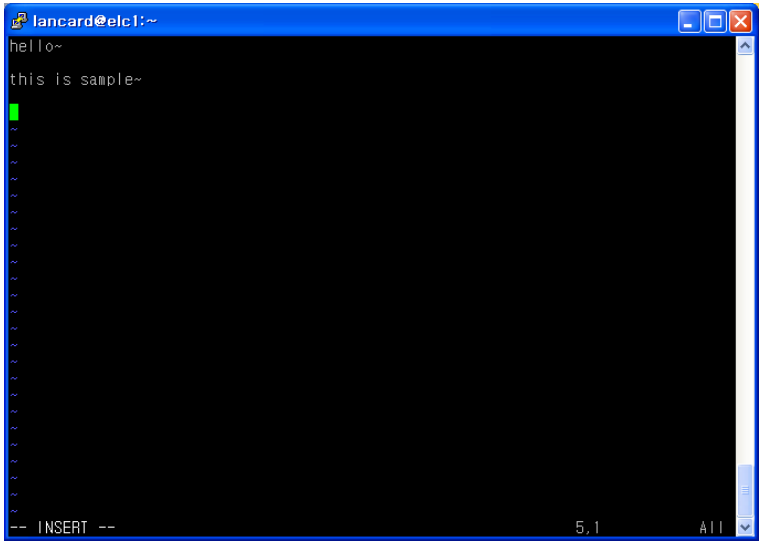
Vi Editor

- Screen shots

(You can see '—INSERT—' when you are in insert mode)



<Command mode>



<Insert mode>

You need help?

- The Linux equivalent of HELP is `man` (manual)
  - Use `man -k <keyword>` to find all commands with that keyword
  - Use `man <command>` to display help for that command
    - Output is presented a page at a time. Use `b` for to scroll backward, `f` or a space to scroll forward and `q` to quit