System Programming

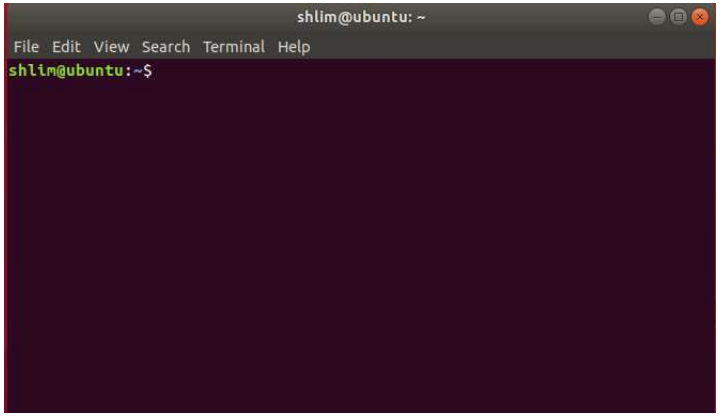
1-3. Linux Shell & Basic Commands

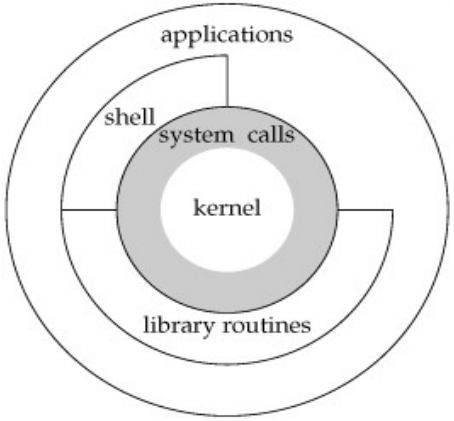
Seung-Ho Lim

Dept. of Computer & Electronic Systems Eng.

Shell?

- The user interface to the operating system
- Functionality:
  - Execute other programs
  - Manage files
  - Manage processes
- A program like any other one
- Executed when you log on





Commonly Used Shells

- /bin/sh The Bourne Shell / POSIX shell
- /bin/csh C shell
- /bin/tcsh Enhanced C Shell
- /bin/ksh Korn shell
- **/bin/bash** Free ksh clone

- Basic form of shell (source):

```
while (read command line from user) {
      parse the command line
      execute the command
}
```

Shell Interactive Use

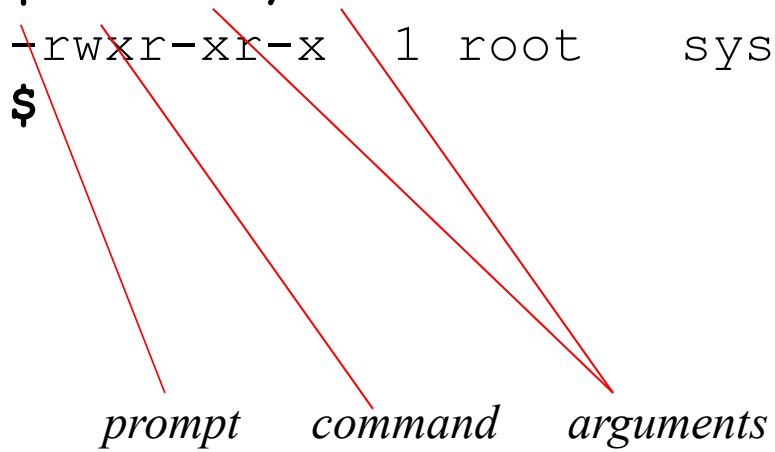
- When you log in, you interactively use the shell:
  - Command history
  - Command line editing
  - File expansion (tab completion support)
  - Command expansion
  - Key bindings
  - Spelling correction
  - Job control

Simple Commands

- ***simple command***: sequence of non blanks arguments separated by blanks or tabs.
- 1st argument (numbered zero) usually specifies the name of the command to be executed.
- Any remaining arguments:
  - Are passed as arguments to that command.
  - Arguments may be filenames, pathnames, directories or special options
  - Special characters are interpreted by shell

Simple Example

```
$ ls -l /bin
-rwxr-xr-x  1 root    sys    43234 Sep 26  2018 date
$
```



- Execute a basic command
- Parsing into command in arguments is called splitting

Types of Arguments

- A command example with several options

```
$ tar –c –v –f archive.tar main.c main.h
```

- Options/Flags
  - convention: -*X* or --*longname*
- Parameters
  - may be files, may be strings
  - depends on command

Frequently Used Commands (1)

- **pwd** print a current working directory
- **cd** change working directory
- **cat** concatenate files and print on the standard output
- **chmod** change a file access permission
- **vi** create/edit files
- **ls** list contents of the current directory
- **rm** remove file
- **mv** rename file
- **cp** copy a file
- **touch** create an empty file
- **mkdir** create a directory
- **rmdir** remove a directory

Frequently Used Commands (2)

- **more** display a file page by page
- **od** display binary files
- **ln** make a link to a file (symbolic or hard link)
- **file** determine file type
- **passwd** change the user password
- **split** split a file

[참고 자료] 리눅스 기본 명령어

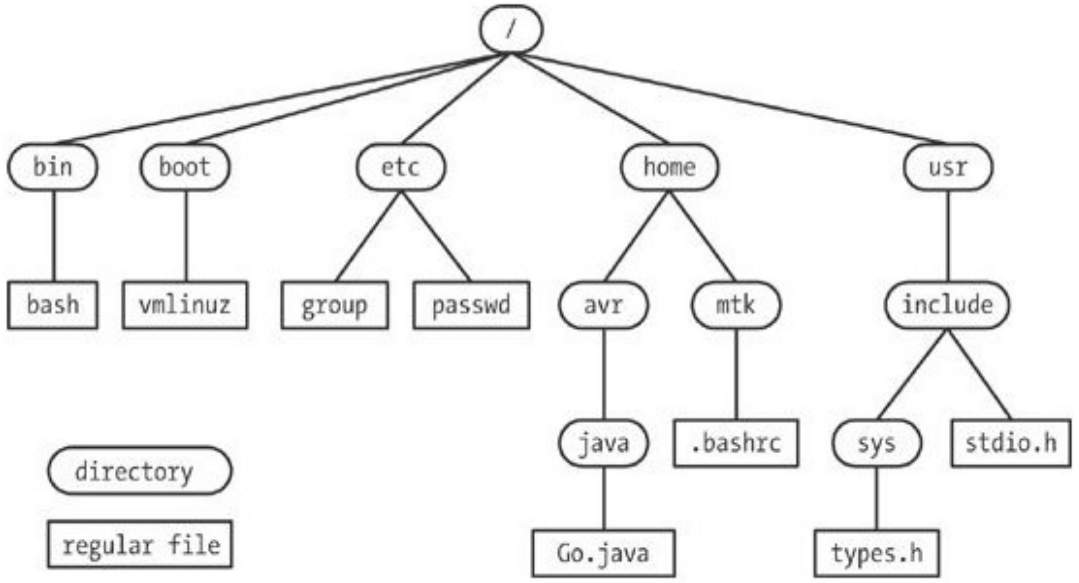
Linux manual sections

- How to use the on-line linux manual pages
  - **man** [section number] [keyword]

    e.g. $ man 3 printf // show the info. on printf in the library section(3)
  - if we omit the section number, man gets it from the first section which have the keyword
- Manual section numbers
  - 1: Executable programs or shell commands
  - 2: System calls
  - 3: Library calls
  - 4: Special files
  - 5. File formats and conventions
  - 6: Games
  - 7: Miscellaneous
  - 8: System administration commands (usually only for root)

Linux File System

- Directory hierarchy



- HOME directory
  - a directory assigned to a user
  - when logging in, the user is first positioned at his/her home!
  - usually placed under `/home`
  - Q: Where is the home for a user '**mtk**' in the above system?

File/Directory Path

- Two types of Path representation
  - Absolute path
    - represent all paths from the root to the position

      e.g. /home/avr/java/Go.java
  - Relative path
    - represent from the current position where I am NOW

    - use single dot '`.`' for the current
    - use double dots '`..`' for the above directory

      e.g. `./java` → a (sub)directory *java* under the current

      e.g. `../include` → a directory *include* in the above of the current

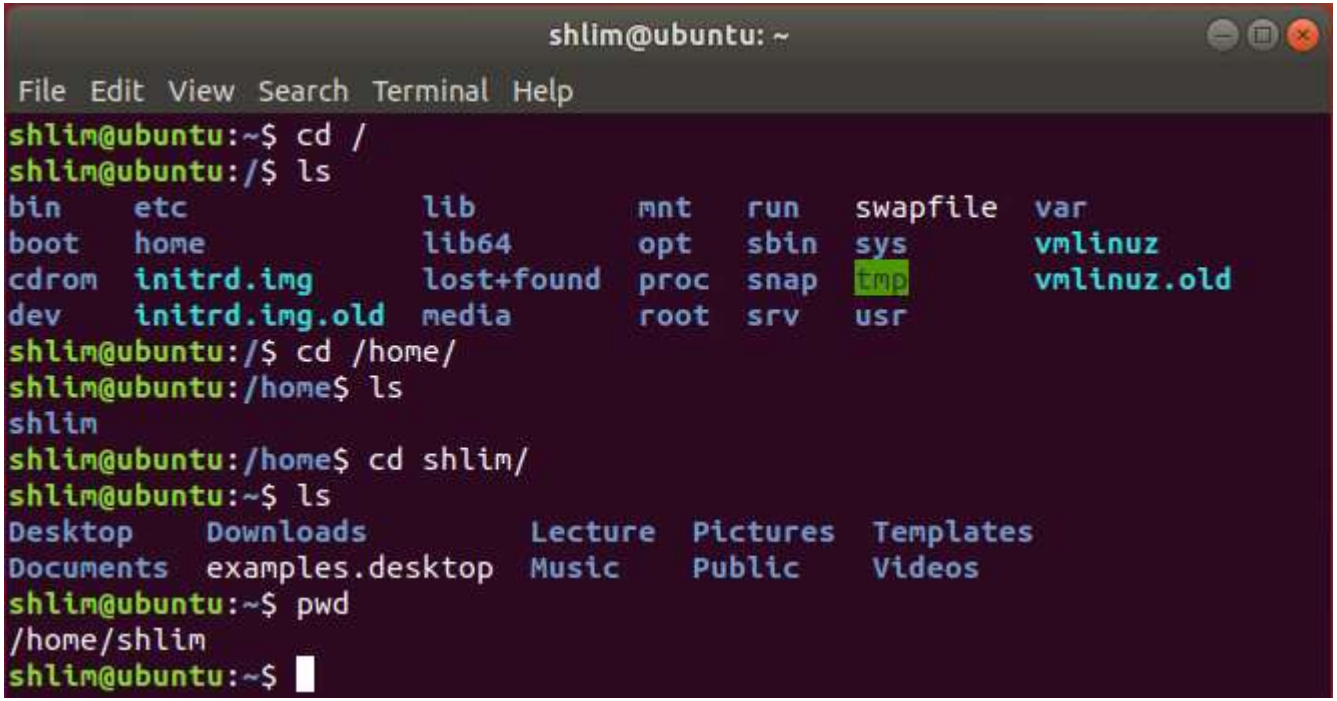
    - use a tild '~' for a user's home

      e.g. cd ~avr → go to the home of user *avr*

      e.g. cd ~ → go to my home

File/Directory Path

- Example

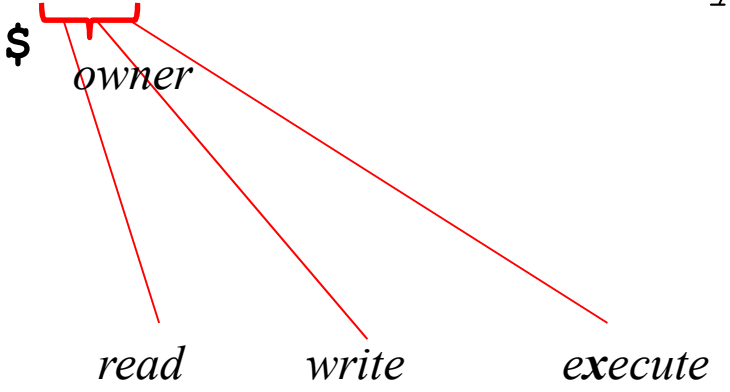


File Permission (1)

- Each file/directory has a permission
  - each file has a owner
  - linux can support a ***group*** of users

    e.g. sp2020 : a group of all students in SP class
  - file/directory specifies a permission to *owner*, *group*, and *others*.

```
$ ls -l /bin
-rwxr-xr-x  1 root   sys    43234 Sep 26  2018 date
$
```



File Permission (2)

- How to change the permission
  - use command : **`chmod`** [permission] [filename]

owner			group			Others		
r	w	x	r	w	x	r	w	x

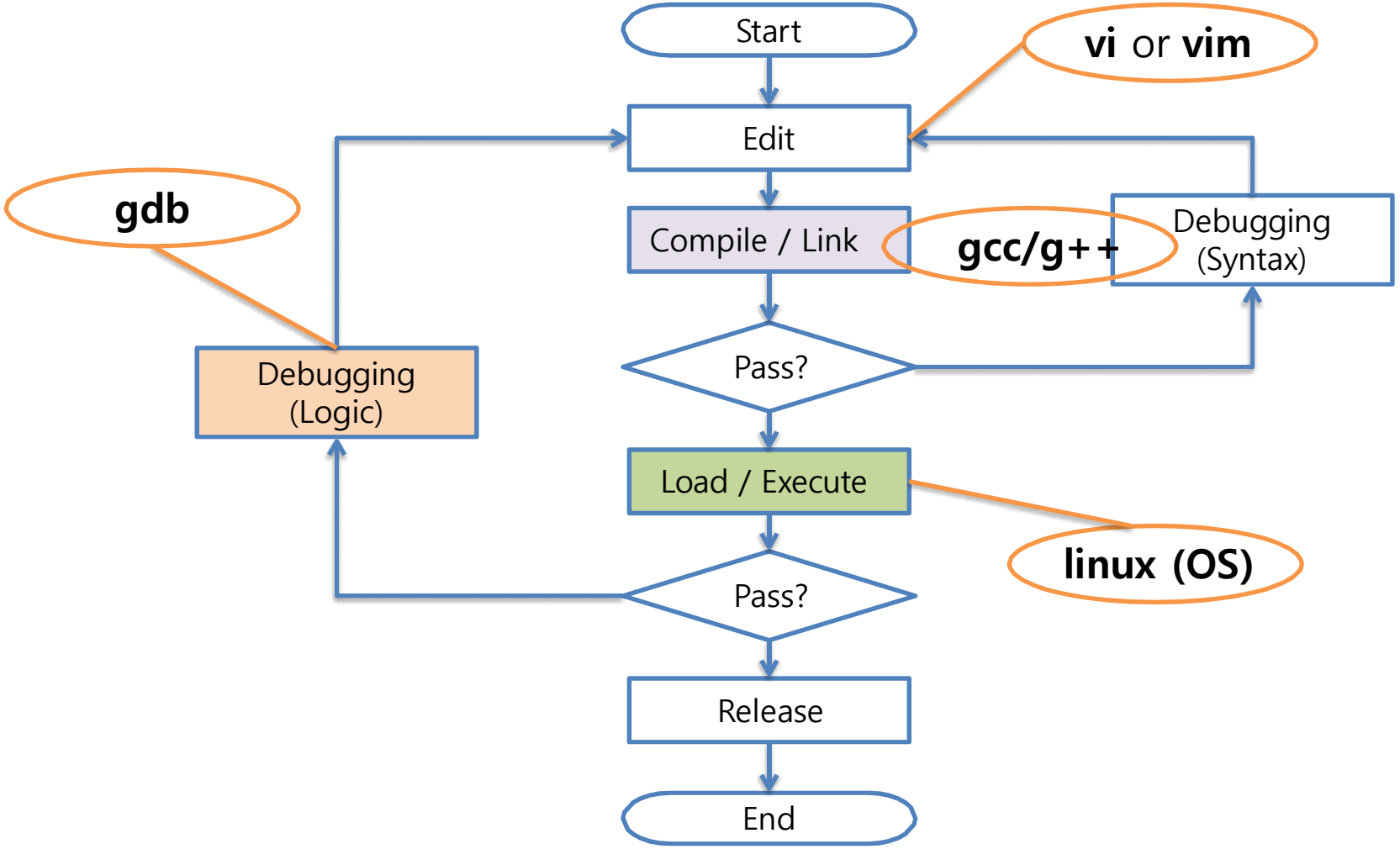
  - permitted if the bit is 1, otherwise NOT permitted
  - e.g.

    For a file ‘myFile.txt’, to give all permission to owner and its group, and only read permission to others

    $111111100_{(2)} = 774_{(8)}$

    $ chmod 774 myFile.txt

Linux Programing Process



gcc Compiler

- Simple examples
  - gcc sample.c → generates a binary *a.out* (by default)
  - ./a.out → execute the a.out

    Note that we should add "**./**" to the filename!
  - gcc sample.c -o sample → generates a binary sample
  - ./sample → execute the binary sample
- Example with multiples sources
  - gcc file1.c file2.c -o fileout

    → compile and link file1.c and file2.c, then generates the *fileout*
  - gcc -c file1.c file2.c

    → compile only and generates the object files
  - gcc –o fileout file1.o file2.o

    → link the file1.o file2.o, then generates the fileout

Homework

- Install XShell onto your computer (notebook)
- Login the linux lab server
- Change your passwd
  - if not changed by next week, the account will be disabled!
- Practice the following linux commands
  - cd, mkdir, rmdir, cp, touch, ls, more, cat