Lecture 2: Shells and Commands

KAIST EE

Linux Shell

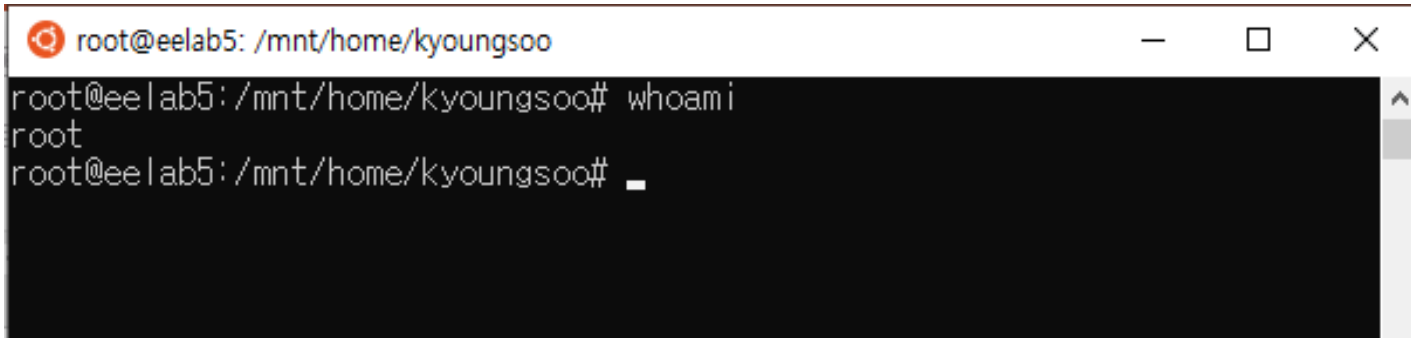
- A program that takes commands and asks the OS to execute them
  - A command line interpreter
  - `sh` (Bourne shell), `bash` (bourne again sh), `ksh`, `csh`/`tcsh`, and so on

```
kyoungsoo@eelab5:~$
```

  - This part is called “prompt” (format: ‘username@servername:directory$’)
  - Can customize the prompt by setting PS1 in .bashrc (https://phoenixnap.com/kb/change-bash-prompt-linux)
  - ‘~’ is the home directory : for user kyoungsoo, ~ = /mnt/home/kyoungsoo/ on eelab5.

Linux Shell

- Super user vs. Normal user
  - root can run all commands
    - e.g., root can run '`shutdown -h now`', which stops the OS and turns it off.
  - Normal users can run only a subset of them.
  - Command `whoami`: get the username.



sudo, su, and man

- `sudo [command]`
  - Superuser-do: runs a command as root

    $ `sudo shutdown –h now`
    - Runs 'shutdown –h now' as root, asks for the root password before running the command
- `su [user-to-be-switched-to]`
  - Switches the current user to *user-to-be-switched-to*

    $ `su yjwon`
    - Switch to user 'yjwon'. It would ask for the password of 'yjwon' before switching

    $ `su`
    - Switch to **root**. It would asks for the root password.
- `man [command]`
  - Shows the manual of *command*

    $ `man sudo` // shows the manual of 'sudo'

Directory Layout

- Knowledge on this allows you to answer.
  - Where are the programs are located?
  - Where do configuration files live?
  - Where can I find log files for this application?
- (Some) common directories
  - `/`: root directory. (not root user) the starting point for all files
  - `/bin`: binaries and other executable programs
  - `/etc`: system configuration files
  - `/home`: home directories
  - `/opt`: optional or third party software
  - `/tmp`: temporary space, typically cleared on reboot
  - `/var`: variable data, most notably log files

Commands Related to Directories/Files

- `ls`: list directory contents

$ `ls -l` // list files with detailed attributes per each file (time, size, permission, etc.)

$ `ls -al` // `-a` means all files

- `cd [dir]`: switches the current directory to `dir`
  - If `dir` is missing, you go to your home directory (~)

$ `cd` // go to the home directory of this user

$ `cd ./temp` // go down to the 'temp' directory under the current directory

- `pwd`: displays the present working directory name
  - It helps if you don't know what directory you're in.
- `cat [files]`: concatenates and displays files

`less/more` are similar commands with slightly different behavior

Commands Related to Directories/Files

- `mkdir/rmdir [dir]`: create/remove [dir]
  - `dir` must be empty before `rmdir`
- `rm [files]`: remove (a) file(s)

$ `rm -r dir` // `-r` option recursively removes all files under `dir` if it is a directory

- `mv [A] [B]`
  - Move file A to file B (rename and/or relocate A)

$ `mv a.txt ~/tmp/` // relocate ~/a.txt to ~/tmp/

$ `mv a.txt b.txt` // rename a.txt to b.txt

$ `mv a.txt ~/tmp/b.txt` // rename and relocate a.txt

$ `mv ./tmp tmp2` // rename a directory, './tmp' to a directory, 'tmp2'

- `cp [A] [B]`
  - Copy file A to file B -> the same as `mv` except it does not remove the original copy [A]

Miscellaneous & Environment Variables

- `echo [argument]`: displays arguments to the screen
- `exit/logout/ctrl-d`: exits the shell or your current session
- `clear`: clears the screen

Environment Variables

- A number of variables that are used to run the program.
- They include
  - `$PATH`: a colon-separated list of directories that the shell searches for commands

$ `echo $PATH //` shows up what's set in the environment variable, 'PATH'

`/bin:/usr/bin:/usr/sbin/:/usr/local/bin` // output of `echo $PATH`

$ `ls`

→ The `bash` shell searches `'ls'` in `/bin` first, and runs it (`/bin/ls`) if it's found. Otherwise, `/usr/bin` will be searched and so on. If no executable command is found in the directories, then the shell says it cannot be found.

Environment Variables

- Other environment variables
  - `$HOME`: the location of the user's home directory.
  - `$PWD`: This variable points to the current directory
  - `$DISPLAY`: the identifier for the display that X11 should use by default.
  - `$LD_LIBRARY_PATH`: a colon-separated list of directories that the dynamic linker should search for shared objects when building a process image after exec, befor e searching in any other directories.

`which` and `-h` option

- `which [command]`: displays the location of command
  - $ `which ls`

    `/bin/ls`
  - $ `which which`

    `/usr/bin/which`
- Hints: -h option
  - Many commands provide hints for how to use them.
- $ cal -h

Working with Directories

- Directory: a container of files and other directories
  - Tree-like structure
  - Special directories
    - `.` : this directory
    - `..` : the parent directory
    - `-` : the previous directory
  - `/` : directory separator. optional for the last directory
  - $ `cd /var/tmp`
    the same as `cd /var/tmp/`
  - $ `cd ..` // go to the parent directory
  - $ `cd -` // go to the previous directory
    - Environment variable, 'OLDPWD' has the previous directory
    - `cd -` is the same as $ `cd $OLDPWD`

```
kyoungsoo@eelab5:~$ pwd
/mnt/home/kyoungsoo
kyoungsoo@eelab5:~$ cd ee415
kyoungsoo@eelab5:~/ee415$ pwd
/mnt/home/kyoungsoo/ee415
kyoungsoo@eelab5:~/ee415$ cd ..
kyoungsoo@eelab5:~$ pwd
/mnt/home/kyoungsoo
kyoungsoo@eelab5:~$ cd ..
kyoungsoo@eelab5:/mnt/home$ pwd
/mnt/home
kyoungsoo@eelab5:/mnt/home$ cd .
kyoungsoo@eelab5:/mnt/home$ pwd
/mnt/home
kyoungsoo@eelab5:/mnt/home$ cd kyoungsoo
kyoungsoo@eelab5:~$ pwd
/mnt/home/kyoungsoo
kyoungsoo@eelab5:~$ echo $OLDPWD
/mnt/home
kyoungsoo@eelab5:~$ cd -
/mnt/home
kyoungsoo@eelab5:/mnt/home$
```

How to Execute a Program in Current Directory?

- Assume you're in your home directory.
  - You have an executable, 'program', in your home directory.
  - Assume $PATH does not have your home directory.
- How to execute the 'program'?
  - $ `program`
    - The shell would complain with 'program: command not found'.
  - $ `./program`
    - '.' represents 'this directory'.
  - $ `~/program`
    - '~' represents my home directory.
  - $ `/mnt/home/kyoungsoo/program`
    - Full path (/mnt/home/kyoungsoo/) of my home directory

Creating/Removing Directories

- Practicing with mkdir/rmdir
  - `-p` : '`mkdir/rmdir -p dir`' creates/removes all intermediate directories in `dir`.

```
kyoungsoo@eelab5:~$ mkdir newdir
kyoungsoo@eelab5:~$ mkdir newdir/product/reviews
mkdir: cannot create directory  'newdir/product/reviews' : No such file or directory
kyoungsoo@eelab5:~$ mkdir -p newdir/product/reviews
kyoungsoo@eelab5:~$ rmdir newdir
rmdir: failed to remove 'newdir': Directory not empty
kyoungsoo@eelab5:~$ rm -rf newdir
kyoungsoo@eelab5:~$ ls newdir
ls: cannot access 'newdir': No such file or directory
kyoungsoo@eelab5:~$ pwd
/mnt/home/kyoungsoo
kyoungsoo@eelab5:~$
```

Listing Files with `ls`

- `$ ls -l`

```
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-rw-r--r-- 1 kyoungsoo Professor 12 Jan 10 23:42 hello.txt
kyoungsoo@eelab5:~/newdir$
```

Permissions

Number of links

Owner name

Group name

File size

Last modified time

File name

- Other options:
  - `-a`: all files including hidden files (e.g., '.bashrc', '.logout')
  - `-F`: displays up types (executables, text files, etc.)
  - `-t`: sort by last modified time (i.e., most recently modified files first)
  - `-d`: displays only directory names not their contents
  - `-R`: list files recursively
  - `-r`: reverse alphabetical order of the file names

Dealing with Spaces in File Names?

- How many files do you see?

```
[kyoungsoo@forest3 newdir]$ ls -l
total 12
-rw-rw-r--. 1 kyoungsoo kyoungsoo  7 Jan 11 17:39 a
-rw-rw-r--. 1 kyoungsoo kyoungsoo  9 Jan 11 17:39 a file
-rw-rw-r--. 1 kyoungsoo kyoungsoo 13 Jan 11 17:39 file
[kyoungsoo@forest3 newdir]$
```

- How to remove the second file, 'a file'?
  - $ `rm a file` // would delete files, 'a' and 'file', *not* 'a file'
  - $ `rm 'a file'` // or `rm "a file"`
  - $ `rm a\ file` // \ (backslash) + space = a character, space: \ is an **escape** char
  - $ `rm a*` // a*: name starting with 'a' followed by zero or more characters
- How to remove a file, '*', below?

```
kyoungsoo@eelab5:~/newdir$ ls -l
total 8
-rw-r--r-- 1 kyoungsoo Professor 8 Jan 11 19:29 *
-rw-r--r-- 1 kyoungsoo Professor 9 Jan 11 19:28 hello
kyoungsoo@eelab5:~/newdir$
```

File permission

- File permission is represented by ten characters.

$$d\underbrace{rwx}_{u}\underbrace{r-x}_{g}\underbrace{rwx}_{o}$$

  - first character: file typ: '-' (regular), 'd' (directory), 'l' (symbolic link: acts as a pointer to another file)
  - Permission: r: readable, w: writable, x: executable
  - Groups: u: users, g: group, o: other
    - User: the user who owns this file
    - Group: users in the same group as the owner
    - Other: all other users
  - Octal representation

$$d\underbrace{r-x}_{5}\underbrace{---}_{0}\underbrace{---}_{0}$$

File and Directory Permissions

```
kyoungsoo@eelab5:~/newdir$ ls -l
total 12
-rw-r--r-- 1 kyoungsoo Professor   12 Jan 11 19:48 file
lrwxrwxrwx 1 kyoungsoo Professor    4 Jan 11 19:49 linktofile -> file
-rwxr-xr-x 1 kyoungsoo Professor   23 Jan 11 19:48 program
drwxr-xr-x 2 kyoungsoo Professor 4096 Jan 11 19:48 subdir
kyoungsoo@eelab5:~/newdir$
```

Change File Permissions

- `chmod [mode] [file]`
  - Changes the modes (or permissions) of a file
  - `mode` format: user-category +-= rwx
    - User-category: u, g, o
    - +: add, -: remove, =: explicitly set the
    - rwx: permissions (read/write/execute)

```
$ chmod g+w file
$ chmod go-x ~
```

- `chmod` supports octal mode (3 values)
  - `$ chmod 640 file //6:rw- 4:r-- 0:---`
    - user group other
  - Each octal number represents 3 bits (r/w/x)
    - E.g., 6 = 110_2 (bit value is 1: "allow," 0: "disallow")
      - readable writable non-executable

```
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-rw-r--r-- 1 kyoungsoo Professor 12 Jan 11 19:48 file
kyoungsoo@eelab5:~/newdir$ clear
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-rw-r--r-- 1 kyoungsoo Professor 12 Jan 11 19:48 file
kyoungsoo@eelab5:~/newdir$ chmod g+w file
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-rw-rw-r-- 1 kyoungsoo Professor 12 Jan 11 19:48 file
kyoungsoo@eelab5:~/newdir$ chmod g-w file
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-rw-r--r-- 1 kyoungsoo Professor 12 Jan 11 19:48 file
kyoungsoo@eelab5:~/newdir$ chmod -w file
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-r--r--r-- 1 kyoungsoo Professor 12 Jan 11 19:48 file
kyoungsoo@eelab5:~/newdir$ chmod ug+x file
kyoungsoo@eelab5:~/newdir$ ls -l
total 4
-r-xr-xr-- 1 kyoungsoo Professor 12 Jan 11 19:48 file
kyoungsoo@eelab5:~/newdir$
```

Default File Creation Mode

- Base permission for a newly created files?
  - 777 for a directory (all can read/write/enter)
  - 666 for a file (all can read and write)
- How to change the behavior above?
  - `umask [-S] [mode]`
    - Sets file creation mode
    - Applied permissions = base permission – [mode]
    - Mode: 3 octal numbers
      - Or symbolic notation with –S option
    - `$ umask 022`
      - Directory: 777 – 022 = 755 (rwxr-xr-x)
      - File: 666 – 022 = 644 (rw-r--r--)
    - -S displays or sets symbolic notation (r/w/x)

The first number: setuid/setgid/sticky
Ignore it for now

```
kyoungsoo@eelab5:~/newdir$ umask
0022
kyoungsoo@eelab5:~/newdir$ umask -S
u=rwx,g=rx,o=rx
kyoungsoo@eelab5:~/newdir$ cat > newfile
hello world
kyoungsoo@eelab5:~/newdir$ ls -l newfile
-rw-r--r-- 1 kyoungsoo Professor 12 Jan 12 18:12 newfile
kyoungsoo@eelab5:~/newdir$ umask 066
kyoungsoo@eelab5:~/newdir$ umask
0066
kyoungsoo@eelab5:~/newdir$ rm newfile
kyoungsoo@eelab5:~/newdir$ cat > newfile
new file with a new file creation mode
kyoungsoo@eelab5:~/newdir$ ls -l newfile
-rw------- 1 kyoungsoo Professor 39 Jan 12 18:13 newfile
kyoungsoo@eelab5:~/newdir$
```

Directory Permissions

```
kyoungsoo@eelab5:~$ ls -ld newdir
drwxr-xr-x 2 kyoungsoo Professor 4096 Jan 13 15:01 newdir
kyoungsoo@eelab5:~$ ls -l newdir
total 16
-rw------- 1 kyoungsoo Professor   39 Jan 12 18:13 newfile
-rwx--x--x 1 kyoungsoo Professor 8512 Jan 13 14:54 prog
kyoungsoo@eelab5:~$ chmod 400 newdir
kyoungsoo@eelab5:~$ ls -ld newdir
dr-------- 2 kyoungsoo Professor 4096 Jan 13 15:01 newdir
kyoungsoo@eelab5:~$ ls -l newdir
ls: cannot access 'newdir/prog': Permission denied
ls: cannot access 'newdir/newfile': Permission denied
total 0
-????????? ? ? ? ?            ? newfile
-????????? ? ? ? ?            ? prog
kyoungsoo@eelab5:~$ newdir/prog
-bash: newdir/prog: Permission denied
kyoungsoo@eelab5:~$ chmod 500 newdir
kyoungsoo@eelab5:~$ ls -ld newdir
dr-x------ 2 kyoungsoo Professor 4096 Jan 13 15:01 newdir
kyoungsoo@eelab5:~$ ls -l newdir
total 16
-rw------- 1 kyoungsoo Professor   39 Jan 12 18:13 newfile
-rwx--x--x 1 kyoungsoo Professor 8512 Jan 13 14:54 prog
kyoungsoo@eelab5:~$ newdir/prog
hello world
kyoungsoo@eelab5:~$
```

- `-d` option: only the directory itself
- `chmod 400`: 4 (100_2 or `r--`)
  - Makes it only readable by the owner
- The next '`ls -l`'
  - Can read the file names
  - But cannot enter the directory
- So, you cannot execute the program under this directory.
- `chmod 500`: 5 (101_2 or `r-x`)
  - Make it readable/enterable
- You can execute newdir/prog.

Finding Files

- `find [path…] [expression…]`
  - Recursively finds files in the path that match the expression
  - No arguments? Find all fines in the current directory
- Examples
  - `find . -name pattern` // `-iname`: case-insensitive

    $ `find /usr/local -name *conf` // file names that end as "conf"

    $ `find . -name "s*" -ls` // run "ls –l" for all matched files
  - `find . -mtime num_days` // files that are num_days old

    $ `find . -mtime +10 -mtime -13` // file age between 10-13 days
  - `find . -size num` // files that are of size num (c (bytes), k (KB), M (MB), G (GB))

    $ `find . -size +300M` // files larger than 300MB
  - `find . -newer fileX` // files that are newer than fileX

    $ `find . -type d -newer b.txt` // directories that are newer than b.txt
  - `find . -exec command {} \;` // run command for each searched file

    $ `find . -exec file {} \;` // run 'file' (it reports the type) for every search file
- If you know file names (or pattern), but don't know the location?
  - $ `locate file-name-pattern` // will show you the path where the file resides

Comparing Files

- `diff [file1] [file2]`
  - Shows the difference between file1 and file2.
  - No output if file1 and file2 are identical.
  - `sdiff` is similar but the output is slightly different.

- '3c3': line 3 (first file) 'c' line 3 (second file)
  - c: line changed
  - d: line deleted
  - a: line added
- '<': first file content, '>': second file content

```
kyoungsoo@eelab5:~$ cat secret
site: facebook.com
user: bob
pass: Abee!
kyoungsoo@eelab5:~$ cat secret.bak
site: facebook.com
user: bob
pass: bee
kyoungsoo@eelab5:~$ diff secret secret.bak
3c3
< pass: Abee!
---
> pass: bee
kyoungsoo@eelab5:~$ sdiff secret secret.bak
site: facebook.com                                      site: facebook.com
user: bob                                               user: bob
pass: Abee!                                           | pass: bee
kyoungsoo@eelab5:~$
```

| : different line

<: exists on only the first file

>: exists on only the second file

Searching in Files

- `grep pattern file`
  - Shows the lines in file that match the pattern
- `grep -v pattern file`
  - Shows the lines in file that **DO NOT** match the pattern
- Other options
  - `-i`: case insensitive
  - `-c`: count the number of occurrences
  - `-n`: precede the output with line numbers in the file

I/O Redirection

- Redirects standard output/input/error of a program to/from a different location
  - A program runs with three standard "files" by default: standard input/output/error
  - A file in unix is an abstract concept: any device that can do input and/or output can be a file
  - Default standard input (file descriptor 0): keyboard
  - Default standard output (file descriptor 1) or error (file descriptor 2): screen
- '>' and '<' symbols
  - `$ prog > fileX` // redirect standard output of prog to fileX
  - `$ prog < fileX` // feed fileX to standard input of prog
  - `$ prog 2> fileX` // redirect standard error of prog to fileX
- Examples
  - `$ ls -l /etc/ > etc.txt` // the output of 'ls –l' is written to a file, etc.txt
  - `$ ls -l /var/ >> etc.txt` // ">>" appends at the end of the file

Pipe

- pipe (`|`) feeds the standard output from the previous command into the standard input of the next command
  - `$ strings music.mp3 | grep -i BTS`
  - `strings`: extract strings from a binary file and prints them out to the standard output
  - The output of strings is fed into the standard input of the next command
- Pipe shows the philosophy of Unix
  - Each program does one thing very well. Pipe allows users to connect multiple programs to flexibly achieve what they want
  - Commands popular for piping: `awk, cat, cut, fmt, join, less, more, n1, pr, sed, seq, sort, tr, and uniq`
  - `$ grep bob passwd | cut -f1,5 -d: | sort | sed 's/:/ /'`
- EE209 students will implement pipe in the final programming assignment

Not All Commands are Covered

- Study these commands for yourself
  - `sort` // sort the input
  - `uniq` // remove repeated lines
  - `gzip` // compress with the zip algorithm (gunzip, zcat)
  - `tar` // save multiple files into one file (or restore multiple files from one file)
  - `awk` // a pattern scanning and processing language
  - `sed` // a stream editor – basic text transformation
- There are many useful tools, but don't need to learn them all at once.

Assignment for Lecture 2

- Deadline: until the start of the next lecture
- Preliminary: prepare hello.c in your home directory
- Please take a snapshot of taking the following steps and upload it to KLMS

1. Create a directory '`hello`' under your home directory
2. Change directory to '`/`' (you should run all commands in this directory)
3. Use gcc209 to compile ~/hello.c and save the output (-o) under `~/hello/` with the name '`hello`'
4. Run the newly created binary (`hello`)
5. See if the binary (`hello`) has a string, "`hello`" in it (use `grep`)
6. Make the directory (`~/hello/`) non-enterable by anybody
7. Show the directory access permission of `~/hello` with '`ls`'
8. Remove the file in `~/hello/` as well as the directory itself.