Open Source SW

Lecture 5
CLI(Command Line Interface)-2

Prof. Youngmin Oh

School of Computing
Gachon University

I/O Redirection: Standard Output

- By default, standard output is screen.
- You can redirect output using “>” after a command (e.g., ls) to create and save the output in a file
- Command “cat” displays the content of a text file.

```
$ pwd
/home/youngmin/OSS/transformers
$ ls -lh
total 16K
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
$ ls -lh > file_list.txt
$ cat file_list.txt
total 16K
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin    0  9월  28 14:17 file_list.txt
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
$
```

I/O Redirection: Standard Output

- Using “>>” appends output to an extising file (if it already exitsts), or create and write to a new file if it doesn’t exist.

```
$ ls -lh >> file_list.txt
$ cat file_list.txt
total 16K
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin    0  9월  28 14:17 file_list.txt
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
total 20K
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin  338  9월  28 14:17 file_list.txt
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
$
```

I/O Redirection: Standard Input

- By default, standard input is from keyboard.
- You can redirecct input from a file using “<”.
- You can mix “<“ and “>” together in a single line.

```
$ cat words.txt
school
class
home
new
lecture
$ sort < words.txt > sorted_words.txt
$ cat sorted_words.txt
class
home
lecture
new
school
$
```

Pipelines “|”

- Pipeline feeds output of previous command to input of next command.
- command1 | command2 | command3 | …

```
$ ls -lh | less
$
```

```
total 28K
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin  676  9월  28 14:23 file_list.txt
:
```

```
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:28 sorted_words.txt
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:27 words.txt
(END)
```

Press “q” key to exit the screen.

```
$ ls | wc -l
7
$
```

Expansion

- Special characters expand its meaning when given to shell commands.

```
$ echo print out the text
print out the text
$
```

```
$ echo *
README.md classification_experiment.py file_list.txt hello_world sorted_
words.txt test.sh words.txt
$
```

```
$ echo ~
/home/youngmin
$ echo ~youngmin
/home/youngmin
$
```

Tip: Backslash

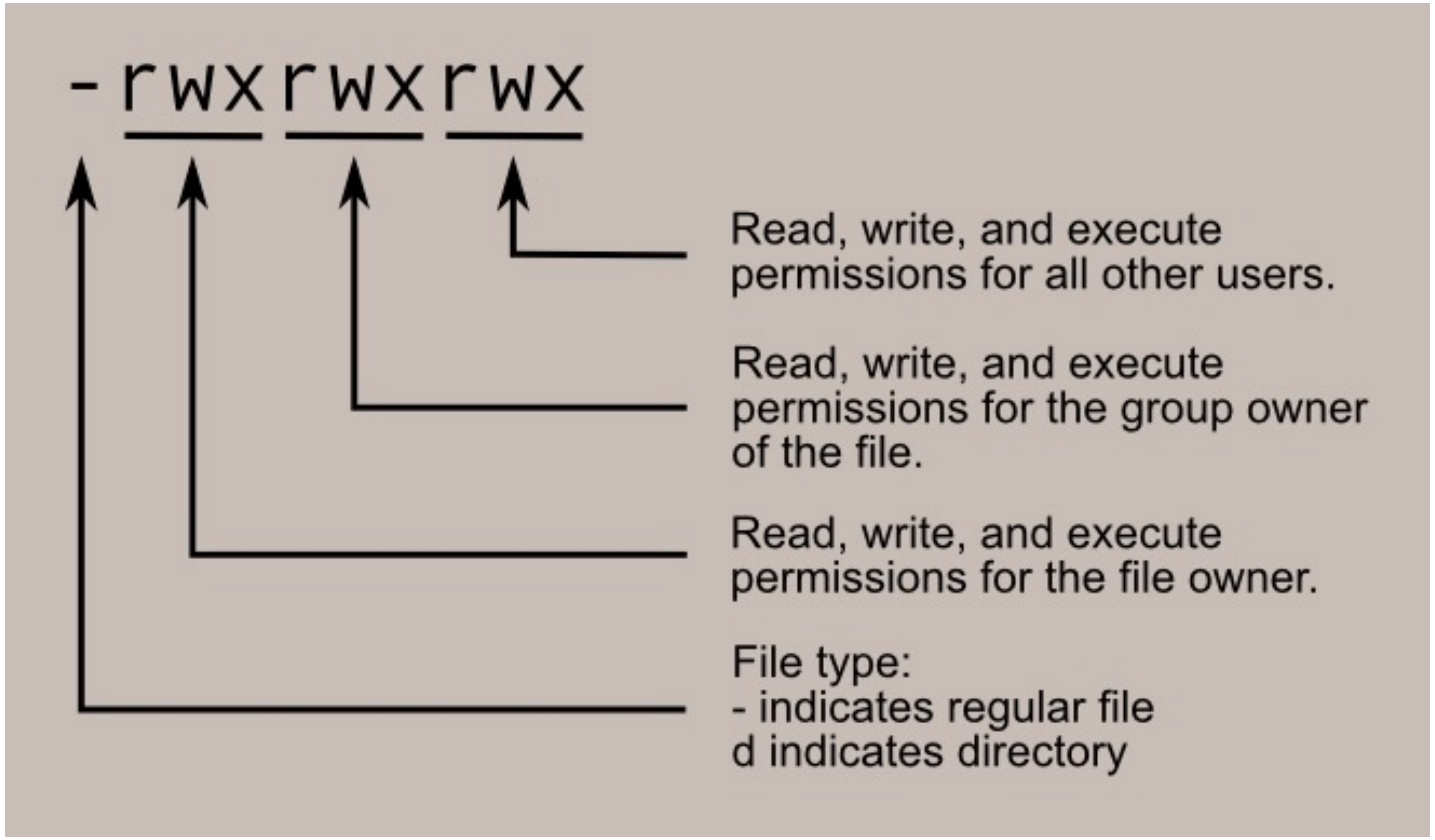
- Backslah can be used to ignore line change in command (“enter”), to enter a long command in multiple lines.

```
$ ls -l \
> --reverse \
> --human-readable
total 28K
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:27 words.txt
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:28 sorted_words.txt
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rw-rw-r-- 1 youngmin youngmin  676  9월  28 14:23 file_list.txt
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
$
```

Permissions

- Linux is a multi-user system.
- Files and directories have a permission assigned differently to owner / group / others.

```
[me@linuxbox me]$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1113504 Jun  6  2019 /bin/bash
```



Source: https://linuxcommand.org/lc3_lts0090.php

Changing Permissions

- "chmod" changes permissions.

```
[me@linuxbox me]$ chmod 600 some_file
```

6 = 110 = rw- for owner
0 = 000 = --- for group
0 = 000 = --- for others

```
rwx rwx rwx = 111 111 111
rw- rw- rw- = 110 110 110
rwx --- --- = 111 000 000

and so on...

rwx = 111 in binary = 7
rw- = 110 in binary = 6
r-x = 101 in binary = 5
r-- = 100 in binary = 4
```

Value	Meaning
777	**(rwxrwxrwx)** No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755	**(rwxr-xr-x)** The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700	**(rwx------)** The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.
666	**(rw-rw-rw-)** All users may read and write the file.
644	**(rw-r--r--)** The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600	**(rw-------)** The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Source: https://linuxcommand.org/lc3_lts0090.php

Changing Permissions

- Change the permission of a file “word.txt” that only the owner (you) can read and write, but all the others (including others in the group) can only read it. No execution is needed for all users.

```
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:27 words.txt
```

```
$ chmod 644 words.txt
$ ls -lh
total 28K
-rw-rw-r-- 1 youngmin youngmin 2.7K  9월  21 18:59 README.md
-rw-rw-r-- 1 youngmin youngmin 3.9K  9월  21 18:59 classification_experiment.py
-rw-rw-r-- 1 youngmin youngmin  676  9월  28 14:23 file_list.txt
-rwxr-xr-x 1 youngmin youngmin   56  9월  28 14:03 hello_world
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:28 sorted_words.txt
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
-rw-r--r-- 1 youngmin youngmin   30  9월  28 14:27 words.txt
$
```

Superuser

- A superuser has all system administation authority.
- Some commands need superuser's privilleges.
- Put "sudo" before the command if you are a superuser.

```
[me@linuxbox me]$ sudo some_command
Password for me:
[me@linuxbox me]$
```

```
[me@linuxbox me]$ sudo -i
Password for me:
root@linuxbox:~#
```

- Type "exit" to get out of a superuser session.

Source: https://linuxcommand.org/lc3_lts0090.php

Text Editors

- In Linux, you can choose CLI-based or GUI-based text editors.

Name	Description	Interface
vi, **vim**	The granddaddy of Unix text editors, **vi**, is infamous for its obtuse user interface. On the bright side, **vi** is powerful, lightweight, and fast. Learning **vi** is a Unix rite of passage, since it is universally available on Unix-like systems. On most Linux distributions, an enhanced version of **vi** called **vim** is provided in place of vi. **vim** is a remarkable editor and well worth taking the time to learn it.	command line
Emacs	The true giant in the world of text editors is Emacs originally written by Richard Stallman. Emacs contains (or can be made to contain) every feature ever conceived of for a text editor. It should be noted that **vi** and Emacs fans fight bitter religious wars over which is better.	command line
nano	**nano** is a free clone of the text editor supplied with the **pine** email program. **nano** is very easy to use but is very short on features compared to **vim** and **emacs**. **nano** is recommended for first-time users who need a command line editor.	command line
gedit	**gedit** is the editor supplied with the GNOME desktop environment. gedit is easy to use and contains enough features to be a good beginners-level editor.	graphical
kwrite	**kwrite** is the "advanced editor" supplied with KDE. It has syntax highlighting, a helpful feature for programmers and script writers.	graphical

Source: https://linuxcommand.org/lc3_wss0010.php

Shell Script

- Write and run a shell script

```
$ nano myscript.sh
```

```
  GNU nano 4.8                          myscript.sh                          Modified
#!/bin/bash

# This is simply a comment line.

echo "Hello Shell Script!"


^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo
```

```
-rw-rw-r-- 1 youngmin youngmin   74  9월  28 15:27 myscript.sh
-rw-rw-r-- 1 youngmin youngmin   30  9월  28 14:28 sorted_words.txt
-rwxrwxr-x 1 youngmin youngmin   23  9월  28 14:13 test.sh
-rw-r--r-- 1 youngmin youngmin   30  9월  28 14:27 words.txt
$ sh myscript.sh
Hello Shell Script!
$
```

Tip: History

- Type “history” to see previous command history.
- Or, save it to a text file.

```
$ history > history_command.txt
$ cat history
```

Lab 5: Lecture Note on Shell Commands

- Make your own lecture note on today's lecture (shell commands)
- There is no predefined structure nor length of note
- Make it help you remember the shell commands
- Use markdown with some markdown formats
- Name it "학번_이름_lecture_note_5.md" and submit to Cyber Campus