

# Open Source SW

## Lecture 5

### CLI(Command Line Interface)-2

JaKeoung Koo  
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  
*다른 곳으로 output 출력 가능.*
- Command “cat” displays the content of a text file.

```
[folder1 $ pwd
/Users/jakeoung/oss/folder1
[folder1 $ ls -lh
total 16
-rw-r--r--  1 jakeoung  staff   13B Jun 26 17:09 README.md
drwxr-xr-x  2 jakeoung  staff   64B Jun 26 18:08 asset
-rw-r--r--  1 jakeoung  staff  2.2K Jun 26 17:09 main.py
drwxr-xr-x  3 jakeoung  staff   96B Jun 26 18:07 new_folder
[folder1 $ ls -lh > file_list.txt
[folder1 $ cat file_list.txt
total 16
-rw-r--r--  1 jakeoung  staff   13B Jun 26 17:09 README.md
drwxr-xr-x  2 jakeoung  staff   64B Jun 26 18:08 asset
-rw-r--r--  1 jakeoung  staff    0B Jun 26 18:59 file_list.txt
-rw-r--r--  1 jakeoung  staff  2.2K Jun 26 17:09 main.py
drwxr-xr-x  3 jakeoung  staff   96B Jun 26 18:07 new_folder
folder1 $
```

*내용 저장*

*← 해당 file로 output이 저장됨.*

*이 file의 내용보.*

# I/O Redirection: Standard Output

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

```
[folder1 $ ls -lh >> file_list.txt
[folder1 $ cat file_list.txt
total 16
-rw-r--r-- 1 jakeoung staff 13B Jun 26 17:09 README.md
drwxr-xr-x 2 jakeoung staff 64B Jun 26 18:08 asset
-rw-r--r-- 1 jakeoung staff 0B Jun 26 18:59 file_list.txt
-rw-r--r-- 1 jakeoung staff 2.2K Jun 26 17:09 main.py
drwxr-xr-x 3 jakeoung staff 96B Jun 26 18:07 new_folder
total 24
-rw-r--r-- 1 jakeoung staff 13B Jun 26 17:09 README.md
drwxr-xr-x 2 jakeoung staff 64B Jun 26 18:08 asset
-rw-r--r-- 1 jakeoung staff 308B Jun 26 18:59 file_list.txt
-rw-r--r-- 1 jakeoung staff 2.2K Jun 26 17:09 main.py
drwxr-xr-x 3 jakeoung staff 96B Jun 26 18:07 new_folder
folder1 $
```

← 기존 file 내용 뒤에 추가.

# I/O Redirection: Standard Input

- By default, standard input is from keyboard.
- You can redirect 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
$ █
```

*INPUT*

*OUTPUT*

# Pipelines “|”

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

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

\$  output을 ↗ Input으로 넘김.

```
total 24
-rw-r--r--  1 jakeoung  staff   13B Jun 26 17:09 README.md
drwxr-xr-x  2 jakeoung  staff   64B Jun 26 18:08 asset
-rw-r--r--  1 jakeoung  staff  616B Jun 26 18:59 file_list.txt
-rw-r--r--  1 jakeoung  staff   2.2K Jun 26 17:09 main.py
drwxr-xr-x  3 jakeoung  staff   96B Jun 26 18:07 new_folder
(END)
```

Press “q” key to exit the screen.

현재 directory 내의 file, directory 개수 셈.

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

5  word counting

# Expansion

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

```
$ echo print out the text print out the text 그대로 출력  
$
```

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

```
folder1 $ echo ~ 현재 directory 주소  
/Users/jakeoung  
folder1 $
```

## Tip: Backslash

- Backslash can be used to ignore line change in command ("enter"), to enter a long command in multiple lines.

```
[folder1 $ ls -l \
[> -h
total 24
-rw-r--r--  1 jakeoung  staff    13B Jun 26 17:09 README.md
drwxr-xr-x  2 jakeoung  staff    64B Jun 26 18:08 asset
-rw-r--r--  1 jakeoung  staff   616B Jun 26 18:59 file_list.txt
-rw-r--r--  1 jakeoung  staff    2.2K Jun 26 17:09 main.py
drwxr-xr-x  3 jakeoung  staff    96B Jun 26 18:07 new_folder
folder1 $
```

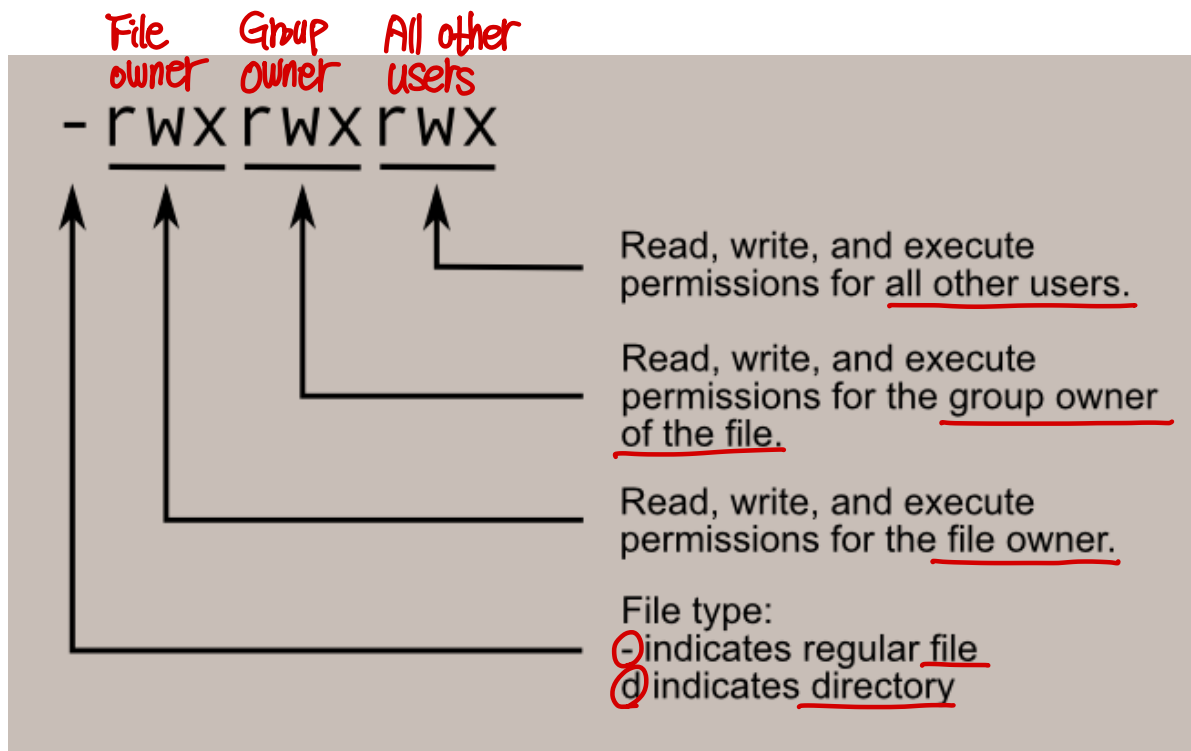
줄바꿈하고 명령어 계속 입력.

# Permissions 권한

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

```
folder1 $ ls -l
total 32
-rw-r--r-- 1 jakeoung staff 13 Jun 26 17:09 README.md
drwxr-xr-x 2 jakeoung staff 64 Jun 26 18:08 asset
```

Handwritten notes above the terminal output:  
- file의 owner (points to 'jakeoung')  
- file이 속한 group (points to 'staff')





# Changing Permissions

- “chmod” changes permissions.

```
folder1 $ chmod 600 README.md
```

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
<b>777</b>	<b>(rwxrwxrwx)</b> No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
<b>755</b>	<b>(rwxr-xr-x)</b> 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.
<b>700</b>	<b>(rwx-----)</b> 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.
<b>666</b>	<b>(rw-rw-rw-)</b> All users may read and write the file.
<b>644</b>	<b>(rw-r--r--)</b> 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.
<b>600</b>	<b>(rw-----)</b> 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.

# 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 jakeung  staff    4 Jun 26 19:34 words.txt
```

110 110 100  
[6] [6] [4] → group 권한 중 writing 배제됨.

```
[folder1 $ chmod 644 words.txt  
[folder1 $ ls -l  
total 32  
-rw-r--r--  1 jakeung  staff    13 Jun 26 17:09 README.md  
drwxr-xr-x  2 jakeung  staff    64 Jun 26 18:08 asset  
-rw-r--r--  1 jakeung  staff   616 Jun 26 18:59 file_list.txt  
-rw-r--r--  1 jakeung  staff  2286 Jun 26 17:09 main.py  
drwxr-xr-x  3 jakeung  staff    96 Jun 26 18:07 new_folder  
-rw-r--r--  1 jakeung  staff     4 Jun 26 19:34 words.txt  
folder1 $
```

변경됨.

# Superuser

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

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

→ Superuser 권한으로 명령 수행

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

일시적으로 superuser가  
되는데 귀찮아서 권장X

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

# Text Editors

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

키보드로만 문서 작업

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

Source: [https://linuxcommand.org/lc3\\_wss0010.php](https://linuxcommand.org/lc3_wss0010.php)

# Shell Script

- Write and run a shell script

```
$ nano myscript.sh
```

새 file 생성

```
GNU nano 4.8 myscript.sh Modified
#!/bin/bash ← bash script 있을 명시.
# This is simply a comment line. ← 주석 (Command 영향 X)
echo "Hello Shell Script!" 새용
```

^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

종료

```
$ sh myscript.sh
Hello Shell Script!
$
```

## Tip

If there is a problem on nano (in Windows), you can edit the file in any other text editor (e.g., 메모장)

## Tip: History

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

*→ 그동안 사용한 명령어들 ↓ txt로 저장.*

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

*내용 출력 ↗*

```
$ cat history_command.txt
```



# wget

- **wget**: download files from the internet directly to your active directory

현재 작업중인

3 (영성)화일 다운로드.

```
(oss) $ wget https://opencv-tutorial.readthedocs.io/en/latest/_images/horse.jpg
--2024-07-05 00:51:18-- https://opencv-tutorial.readthedocs.io/en/latest/_image
s/horse.jpg
Resolving opencv-tutorial.readthedocs.io (opencv-tutorial.readthedocs.io)... 104
.16.253.120, 104.16.254.120
Connecting to opencv-tutorial.readthedocs.io (opencv-tutorial.readthedocs.io)|10
4.16.253.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 110487 (108K) [image/jpeg]
Saving to: 'horse.jpg.1'

horse.jpg.1          100%[=====>] 107.90K  --.-KB/s    in 0.002s

2024-07-05 00:51:18 (52.3 MB/s) - 'horse.jpg.1' saved [110487/110487]
```

# curl

- curl: fetching, uploading, and managing data over the Internet  
curl [options] [URL]

```
(oss) $ curl -o horse.jpg https://opencv-tutorial.readthedocs.io/en/latest/_images/horse.jpg
```

다운시 저장할 파일명

```
(oss) $ curl -O https://opencv-tutorial.readthedocs.io/en/latest/_images/horse.jpg
```



# grep

- `grep` (Global Regular Expression Print) for searching text within files. *파일에서 text 검색.*
- `grep "search_term" file.txt`
  - Searches for the exact "search\_term" within "file.txt" and prints matching lines. *해당 line 출력.*
- **\*\*Common Options:\*\***
  - `-i`: Case-insensitive search (finds "apple" and "Apple").
  - `-v`: Invert the match (finds lines *\*not\** containing the search term). *없을때 출력.*
  - `-n`: Display line numbers along with matching lines. *해당 line number도 출력*
  - `-r`: Recursive search (searches through all files in a directory and its subdirectories). *현재+하위까지 찾음.*
- `grep` supports powerful regular expressions for more complex searches.
  - `.`: Matches any character (`^.*`) zero or more times (`^*`).
  - `[d]`: Matches any digit (0-9).
  - `[abc]`: Matches any single character within the brackets.
  - `^`: Matches the beginning of a line.
  - `$`: Matches the end of a line.

*이 문장에 test가 있는지.*

```
(oss) $ echo "This is a test sentence with some words." | grep "test"
This is a test sentence with some words.
```

```
[(oss) $ echo "This is a test sentence with some words." | grep "Test"
(oss) $ echo "This is a test sentence with some words." | grep -i "Test"
This is a test sentence with some words.
```

*대소구분X*

```
(oss) $ echo "This is a test sentence with some words." | grep -n ^[A-Z]
1:This is a test sentence with some words.
```

*A부터 Z까지 후 첫단어에 존재시 line num*

```
(oss) $ echo "This is a test sentence with some words." | grep -i ^t
This is a test sentence with some words.
```

*t로 시작 + 대소구분X*

# If you need help, ask to LLM!

J I want to remove the files whose names start with abc and end with !. Can you give a shell command for that?

Here's a shell command to remove files whose names start with "abc" and end with "!":

```
bash
rm abc*!
```

This command uses the wildcard `*` to match any characters between "abc" and "!".

Would you like me to explain how this command works or provide any additional information?

⚠ Claude does not have the ability to run the code it generates yet.

J Yes, please explain.

Certainly. Let's break down the command:

```
bash
rm abc*!
```

1. `rm`: This is the Unix/Linux command for removing (deleting) files and directories.
2. `abc`: This matches the beginning of the filename exactly.
3. `*`: This is a wildcard character that matches any number of characters (including zero characters). It allows for any content between "abc" and "!".
4. `!`: This matches the end of the filename exactly.

claude.ai

# 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
- For the submission, check Cybercampus