

시스템프로그래밍실습 보고서

Basic 과제

과목	시스템프로그래밍실습
담당교수	이기훈교수님
학과	컴퓨터정보공학부
학번	2021202058
이름	송채영

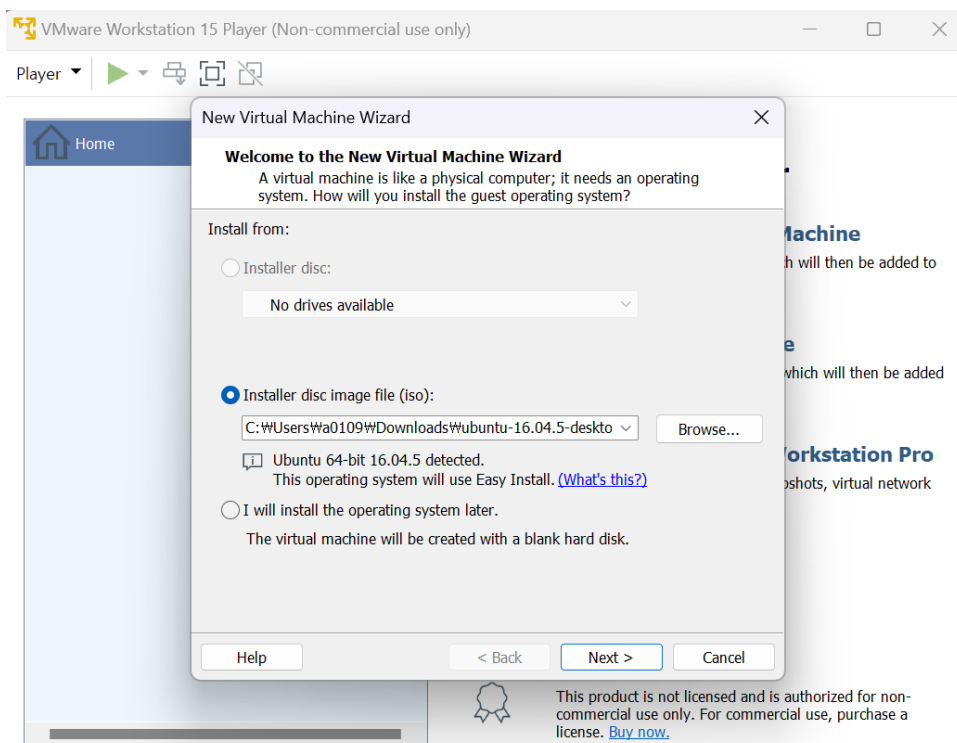
1. Introduction

Vmware 와 ubuntu 를 설치하는 과정을 캡처하고 설치하는 과정에 대한 설명을 작성한다. 설치한 프로그램을 통해 실습시간에 배웠던 Linux 명령어를 사용하고 이를 캡처하여 간단한 설명을 작성해본다. 더 나아가 각각의 명령어들이 수행하는 역할, 즉 명령어의 핵심 내용에 대해 설명한다. 마지막으로 Vi editor 를 사용하는 방법과 단축키를 익혀본다. 또한 Make 를 사용하는 방법을 익히고 컴파일하고 디버그를 진행해본다.

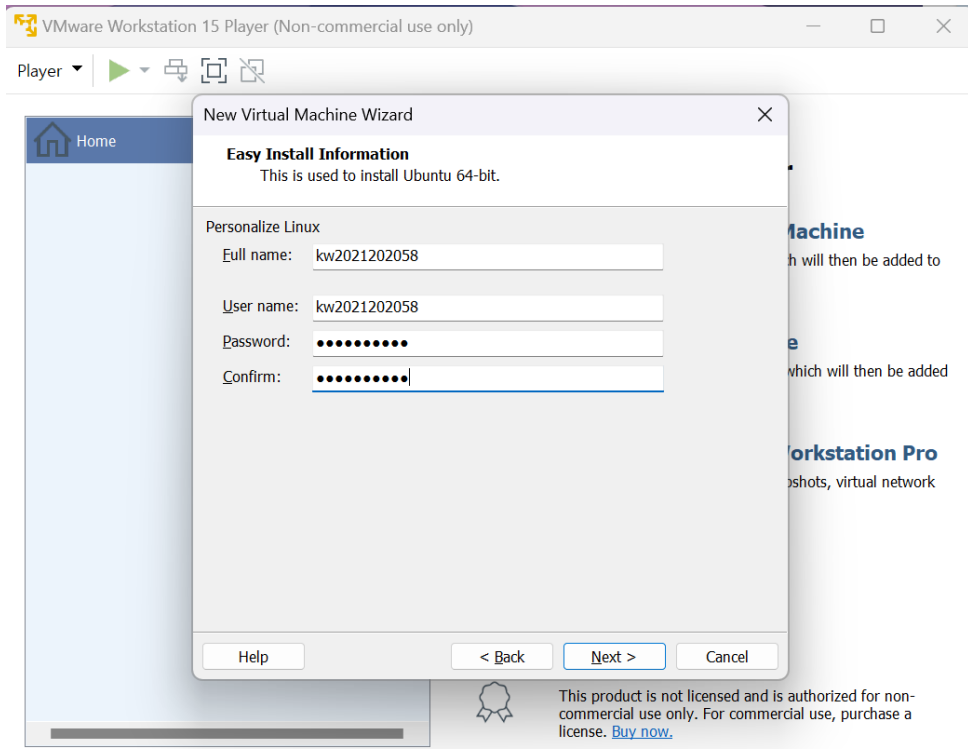
2. 결과 화면

2-1. Ubuntu Installation

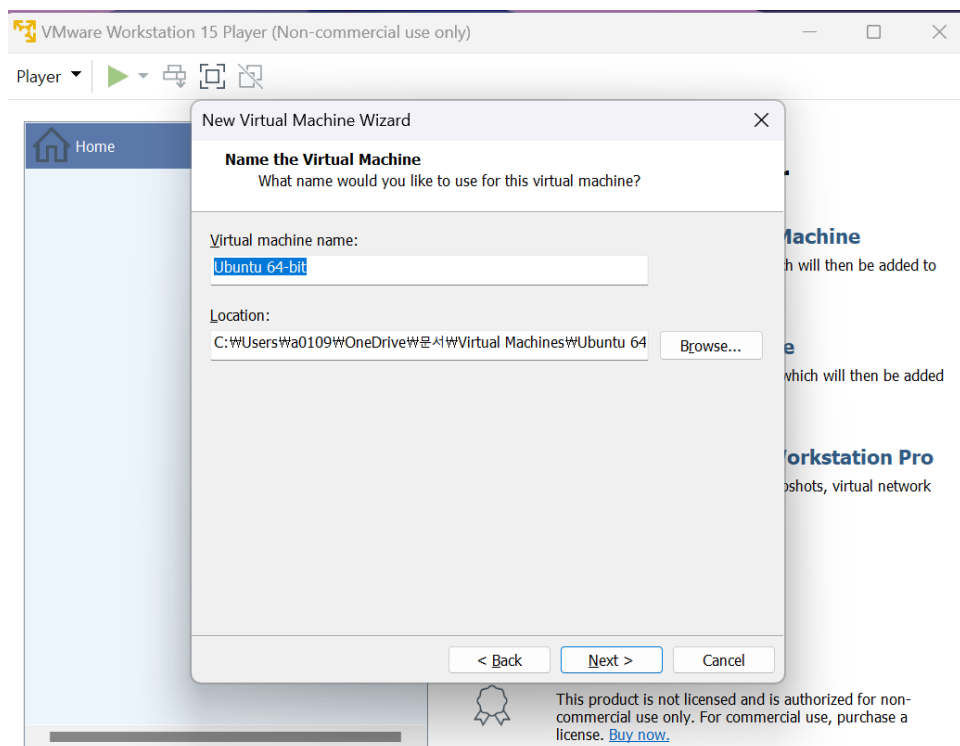
Ubuntu 16.04.5 와 Vmware 15.5.7 을 설치하였다. 옵션 선택은 기본 값을 유지하였다. 설치 과정은 과제에서 제외이므로 Vmware 의 설치과정은 생략한다.



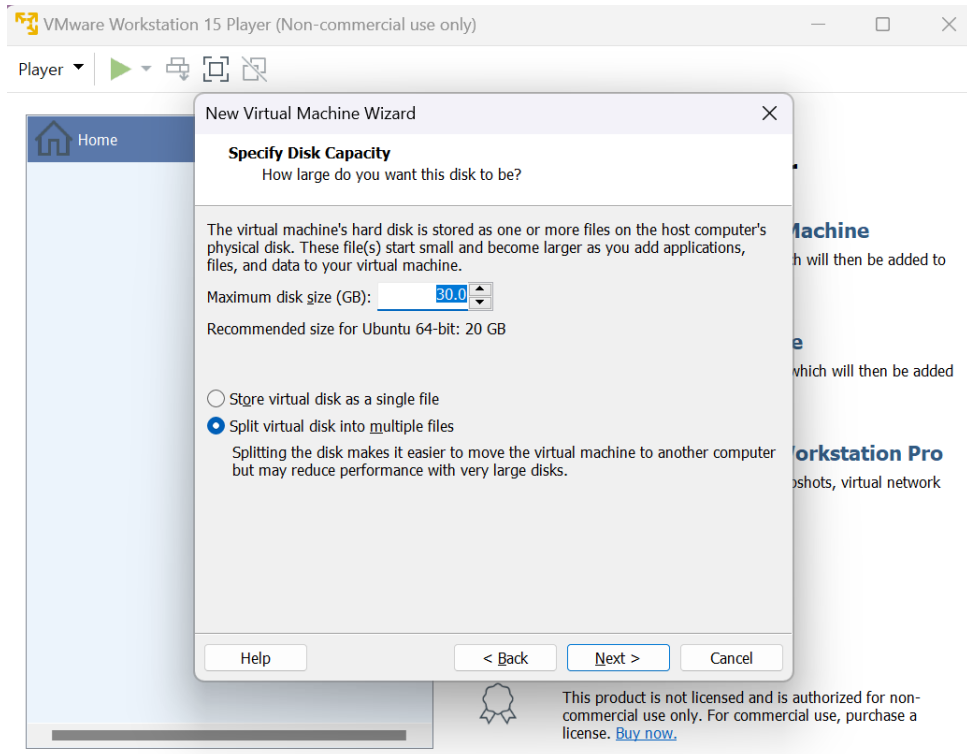
가상머신을 생성하는 과정으로, VMware 를 이용해 Ubuntu 를 설치하였다. 위의 사진은 Ubuntu ISO 파일 위치를 지정해주는 과정을 캡처한것이다.



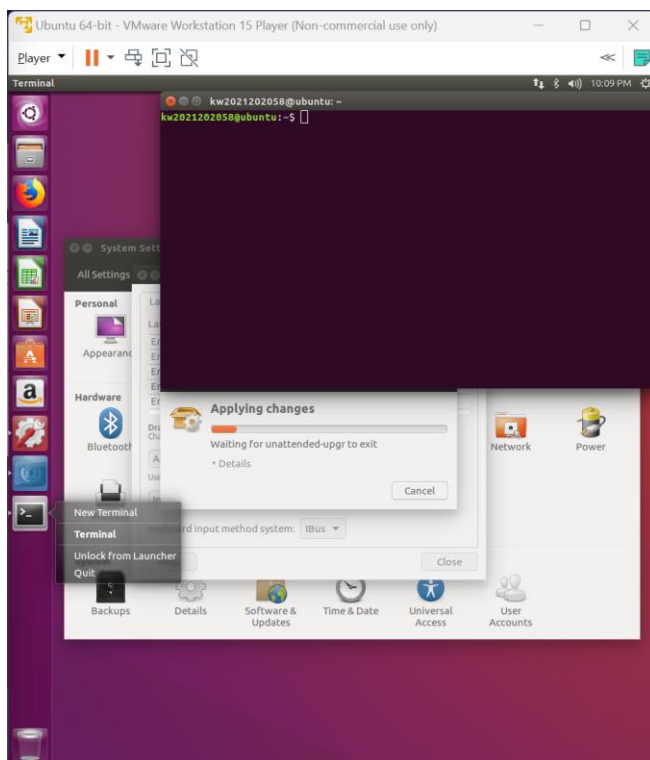
위의 사진은 Ubuntu 에서 로그인 할 계정을 설정하기 위해 User name 인 ID 와 비밀번호인 Pssword 와 Confirm 을 지정해주었다. 과제에서 계정 ID 는 kw+학번으로 지정되어 있기 때문에 kw2021202058 로 설정해주었다.



위 사진은 가상머신의 이름과 설치될 위치를 지정해주는 과정을 캡처한것이다.



가상머신에 할당할 저장장치의 크기를 설정해주는 과정을 캡처한 사진이다. 가상머신에 할당할 메인 메모리의 크기와 cpu 그리고 cpu core 의 수를 설정해주는 과정이다. Disk Size 는 30 으로 설정해주었다.



모든 설치과정을 거친 후 terminal 를 dock 에 고정하였다. Terminal 에 사용자 명이 kw2021202058 로 지정되어 있는 것을 확인할 수 있다.

2-2. Linux Commands

- preparation

```
kw2021202058@ubuntu: ~  
kw2021202058@ubuntu:~$ ls  
Assignment1 Desktop examples.desktop Pictures srv Videos  
cli Documents makefile Public srv.c  
cli.c Downloads Music splab_commands Templates  
kw2021202058@ubuntu:~$ ls -l  
total 88  
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:50 Assignment1  
-rwxrwxr-x 1 kw2021202058 kw2021202058 8600 Mar 16 23:58 cli  
-rw-rw-r-- 1 kw2021202058 kw2021202058 67 Mar 16 23:53 cli.c  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:10 Desktop  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Documents  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Downloads  
-rw-r--r-- 1 kw2021202058 kw2021202058 8980 Mar 16 23:46 examples.desktop  
-rw-rw-r-- 1 kw2021202058 kw2021202058 72 Mar 16 23:59 makefile  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Music  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Pictures  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Public  
-rwxrwxr-x 1 kw2021202058 kw2021202058 2690 Feb 26 20:30 splab_commands  
-rwxrwxr-x 1 kw2021202058 kw2021202058 8600 Mar 16 23:59 srv  
-rw-rw-r-- 1 kw2021202058 kw2021202058 66 Mar 16 23:54 srv.c  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Templates  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Videos  
kw2021202058@ubuntu:~$ chmod +x splab_commands  
kw2021202058@ubuntu:~$ ./splab_commands  
kw2021202058@ubuntu:~$ ls -l  
total 92  
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:50 Assignment1  
-rwxrwxr-x 1 kw2021202058 kw2021202058 8600 Mar 16 23:58 cli  
-rw-rw-r-- 1 kw2021202058 kw2021202058 67 Mar 16 23:53 cli.c  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:10 Desktop  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Documents  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Downloads  
-rw-r--r-- 1 kw2021202058 kw2021202058 8980 Mar 16 23:46 examples.desktop  
-rw-rw-r-- 1 kw2021202058 kw2021202058 72 Mar 16 23:59 makefile  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Music  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Pictures  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Public  
-rwxrwxr-x 1 kw2021202058 kw2021202058 2690 Feb 26 20:30 splab_commands  
-rwxrwxr-x 1 kw2021202058 kw2021202058 8600 Mar 16 23:59 srv  
-rw-rw-r-- 1 kw2021202058 kw2021202058 66 Mar 16 23:54 srv.c  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Templates  
drwxr-xr-x 2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Videos  
drwxrwxr-x 3 kw2021202058 kw2021202058 4096 Mar 19 23:12 work  
kw2021202058@ubuntu:~$
```

실습 자료와 같이 올려 주신 'splab_commands' 파일을 실행해주어 work 파일이 생성된 것을 확인할 수 있다.

- man

man 명령어를 사용하여 해당 명령어의 매뉴얼(이름, 사용 형식, 개요)과 프로그램의 사용법을 확인할 수 있다. man 명령어는 man 다음에 찾고 싶은 명령어를 입력하여 사용한다.

- man ls

```
kw2021202058@ubuntu: ~
LS(1)                                User Commands                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default).
  Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
  fied.

  Mandatory arguments to long options are mandatory for short options
  too.

  -a, --all
      do not ignore entries starting with .

  -A, --almost-all
      do not list implied . and ..

  --author
      with -l, print the author of each file

  -b, --escape
      print C-style escapes for nongraphic characters

  --block-size=SIZE
      scale sizes by SIZE before printing them; e.g., '--block-size=M'
      prints sizes in units of 1,048,576 bytes; see SIZE format below

  -B, --ignore-backups
      do not list implied entries ending with ~

  -c      with -lt: sort by, and show, ctime (time of last modification of
          file status information); with -l: show ctime and sort by name;
          otherwise: sort by ctime, newest first

  -C      list entries by columns

  --color[=WHEN]
      colorize the output; WHEN can be 'always' (default if omitted),
      'auto', or 'never'; more info below

  -d, --directory
      list directories themselves, not their contents

  -D, --dired
      generate output designed for Emacs' dired mode

  -f      do not sort, enable -aU, disable -ls --color

Manual page ls(1) line 1 (press h for help or q to quit)
```

위 사진과 같이 man ls 를 입력 시 ls 명령어에 대한 매뉴얼을 보여준다.

- man -k copy

```

kw2021202058@ubuntu:~$ man -k copy
Clone (3pm) - recursively copy Perl datatypes
bcopy (3) - copy byte sequence
copysign (3) - copy sign of a number
copysignf (3) - copy sign of a number
copysignl (3) - copy sign of a number
cp (1) - copy files and directories
cpgr (8) - copy with locking the given file to the password or gr...
cpio (1) - copy files to and from archives
cppw (8) - copy with locking the given file to the password or gr...
dd (1) - convert and copy a file
debconf-copydb (1) - copy a debconf database
File::Copy::Recursive (3pm) - Perl extension for recursively copying files an...
getunwind (2) - copy the unwind data to caller's buffer
getutmp (3) - copy utmp structure to utmpx, and vice versa
getutmpx (3) - copy utmp structure to utmpx, and vice versa
gvfs-copy (1) - Copy files
gvfs-move (1) - Copy files
install (1) - copy files and set attributes
mcopy (1) - copy MSDOS files to/from Unix
memccpy (3) - copy memory area
memcpy (3) - copy memory area
memmove (3) - copy memory area
mempcpy (3) - copy memory area
ntfscpy (8) - copy file to an NTFS volume.
objcopy (1) - copy and translate object files
rcp (1) - secure copy (remote file copy program)
rsync (1) - a fast, versatile, remote (and local) file-copying tool
scp (1) - secure copy (remote file copy program)
ssh-copy-id (1) - use locally available keys to authorise logins on a re...
stpcpy (3) - copy a string returning a pointer to its end
stpncpy (3) - copy a fixed-size string, returning a pointer to its end
strcpy (3) - copy a string
strncpy (3) - copy a string
va_copy (3) - variable argument lists
wcpcpy (3) - copy a wide-character string, returning a pointer to i...
wcpncpy (3) - copy a fixed-size string of wide characters, returning...
wcscpy (3) - copy a wide-character string
wcsncpy (3) - copy a fixed-size string of wide characters
wmemcpy (3) - copy an array of wide-characters
wmemmove (3) - copy an array of wide-characters
wmemcpy (3) - copy memory area
x86_64-linux-gnu-objcopy (1) - copy and translate object files
kw2021202058@ubuntu:~$

```

위 사진과 같이 `man -k copy` 를 입력 시 `copy` 단어가 포함된 모든 명령어를 보여준다.

- `man -a write`

```
kw2021202058@ubuntu: ~  
WRITE(1) BSD General Commands Manual WRITE(1)  
  
NAME  
    write - send a message to another user  
  
SYNOPSIS  
    write user [tty]  
  
DESCRIPTION  
    The write utility allows you to communicate with other users, by copying lines from your terminal to theirs.  
  
    When you run the write command, the user you are writing to gets a message of the form:  
  
        Message from yourname@yourhost on yourtty at hh:mm ...  
  
    Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, they must run write as well.  
  
    When you are done, type an end-of-file or interrupt character. The other user will see the message 'EOF' indicating that the conversation is over.  
  
    You can prevent people (other than the super-user) from writing to you with the mesg(1) command.  
  
    If the user you want to write to is logged in on more than one terminal, you can specify which terminal to write to by specifying the terminal name as the second operand to the write command. Alternatively, you can let write select one of the terminals - it will pick the one with the shortest idle time. This is so that if the user is logged in at work and also dialed up from home, the message will go to the right place.  
  
    The traditional protocol for writing to someone is that the string '-o', either at the end of a line or on a line by itself, means that it is the other person's turn to talk. The string 'oo' means that the person believes the conversation to be over.  
  
SEE ALSO  
    mesg(1), talk(1), wall(1), who(1)  
  
HISTORY  
    A write command appeared in Version 1 AT&T UNIX.  
  
BUGS  
    The sender's LC_CTYPE setting is used to determine which characters are safe to write to a terminal, not the receiver's (which write has no way of knowing).  
  
    The write utility does not recognize multibyte characters.  
  
BSD July 17, 2004 BSD  
Manual page write(1) line 1 (press h for help or q to quit)
```

위 사진과 같이 `man -a write` 를 입력 시 write 명령어의 모든 섹션을 보여준다. `man write` 는 섹션 중 가장 하위 섹션이 나온다는 점에서 차이점이 있다.

- `man kill`


```
kw2021202058@ubuntu: ~
KILL(1) User Commands KILL(1)

NAME
    kill - send a signal to a process

SYNOPSIS
    kill [options] <pid> [...]

DESCRIPTION
    The default signal for kill is TERM. Use -l or -L to list available
    signals. Particularly useful signals include HUP, INT, KILL, STOP,
    CONT, and 0. Alternate signals may be specified in three ways: -9,
    -SIGKILL or -KILL. Negative PID values may be used to choose whole
    process groups; see the PGID column in ps command output. A PID of -1
    is special; it indicates all processes except the kill process itself
    and init.

OPTIONS
    <pid> [...]
        Send signal to every <pid> listed.

    -<signal>
    -s <signal>
    --signal <signal>
        Specify the signal to be sent. The signal can be specified by
        using name or number. The behavior of signals is explained in
        signal(7) manual page.

    -l, --list [signal]
        List signal names. This option has optional argument, which
        will convert signal number to signal name, or other way round.

    -L, --table
        List signal names in a nice table.

NOTES
    Your shell (command line interpreter) may have a built-in kill
    command. You may need to run the command described here as
    /bin/kill to solve the conflict.

EXAMPLES
    kill -9 -1
        Kill all processes you can kill.

    kill -l 11
        Translate number 11 into a signal name.

    kill -L
        List the available signal choices in a nice table.

    kill 123 543 2341 3453
        Send the default signal, SIGTERM, to all those processes.

Manual page kill(1) line 1 (press h for help or q to quit)
```

위 사진과 같이 `man kill` 을 입력 시 `kill` 명령어에 대한 매뉴얼을 보여준다.

- ls

```

kw2021202058@ubuntu:~$ ls
Assignment1 Desktop examples.desktop Pictures srv Videos
cli Documents makefile Public srv.c work
cli.c Downloads Music splab_commands Templates
kw2021202058@ubuntu:~$ ls -a
. Downloads srv
.. examples.desktop srv.c
Assignment1 .gconf .sudo_as_admin_successful
.bash_history .gnupg Templates
.bash_logout .ICEauthority Videos
.bashrc .lessht .viminfo
.cache .local work
cli makefile .Xauthority
cli.c Music .xinputrc
.config Pictures .xsession-errors
Desktop .profile .xsession-errors.old
.dmrc Public
Documents splab_commands
kw2021202058@ubuntu:~$ ls -F
Assignment1/ Documents/ Music/ srv* work/
cli* Downloads/ Pictures/ srv.c
cli.c examples.desktop Public/ Templates/
Desktop/ makefile splab_commands* Videos/

```

```

kw2021202058@ubuntu:~$ ls -al
total 168
drwxr-xr-x 17 kw2021202058 kw2021202058 4096 Mar 19 23:25 .
drwxr-xr-x  3 root          root          4096 Mar 16 23:46 ..
drwxrwxr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:50 Assignment1
-rw-r--r--  1 kw2021202058 kw2021202058  527 Mar 19 23:10 .bash_history
-rw-r--r--  1 kw2021202058 kw2021202058  220 Mar 16 23:46 .bash_logout
-rw-r--r--  1 kw2021202058 kw2021202058 3771 Mar 16 23:46 .bashrc
drwx----- 14 kw2021202058 kw2021202058 4096 Mar 19 23:10 .cache
-rwxrwxr-x  1 kw2021202058 kw2021202058 8600 Mar 16 23:58 cli
-rw-rw-r--  1 kw2021202058 kw2021202058   67 Mar 16 23:53 cli.c
drwx----- 14 kw2021202058 kw2021202058 4096 Mar 16 23:50 .config
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 19 23:10 Desktop
-rw-r--r--  1 kw2021202058 kw2021202058   25 Mar 16 23:49 .dmrc
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Documents
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Downloads
-rw-r--r--  1 kw2021202058 kw2021202058 8980 Mar 16 23:46 examples.desktop
drwx-----  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 .gconf
drwx-----  3 kw2021202058 kw2021202058 4096 Mar 19 23:08 .gnupg
-rw-r--r--  1 kw2021202058 kw2021202058 1908 Mar 19 23:08 .ICEauthority
-rw-r--r--  1 kw2021202058 kw2021202058   28 Mar 19 23:25 .lessht
drwx-----  3 kw2021202058 kw2021202058 4096 Mar 16 23:49 .local
-rw-rw-r--  1 kw2021202058 kw2021202058   72 Mar 16 23:59 makefile
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Music
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Pictures
-rw-r--r--  1 kw2021202058 kw2021202058  655 Mar 16 23:46 .profile
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Public
-rwxrwxrwx  1 kw2021202058 kw2021202058 2690 Feb 26 20:30 splab_commands
-rwxrwxr-x  1 kw2021202058 kw2021202058 8600 Mar 16 23:59 srv
-rw-rw-r--  1 kw2021202058 kw2021202058   66 Mar 16 23:54 srv.c
-rw-r--r--  1 kw2021202058 kw2021202058    0 Mar 16 23:51 .sudo_as_admin_success
ful
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Templates
drwxr-xr-x  2 kw2021202058 kw2021202058 4096 Mar 16 23:49 Videos
-rw-r--r--  1 kw2021202058 kw2021202058 2539 Mar 17 00:00 .viminfo
drwxrwxr-x  3 kw2021202058 kw2021202058 4096 Mar 19 23:12 work
-rw-r--r--  1 kw2021202058 kw2021202058   51 Mar 19 23:08 .Xauthority
-rw-rw-r--  1 kw2021202058 kw2021202058  131 Mar 18 22:32 .xinputrc
-rw-r--r--  1 kw2021202058 kw2021202058   82 Mar 19 23:08 .xsession-errors
-rw-r--r--  1 kw2021202058 kw2021202058 1173 Mar 19 07:35 .xsession-errors.old

```

ls 명령어는 폴더 안에 있는 파일을 보여준다. option 에 따라서 ls -a 를 입력 시 숨긴 파일을 포함해서 모든 파일을 출력한다. ls -F 를 입력 시 파일 종류를 표시해준다. '/'는

directory 를 의미하며, '*'는 실행파일을 의미한다. ls -l 를 입력 시 파일 정보를 자세하게 출력한다. 두 번째 사진에서는 ls -al 을 입력해주었는데, 숨겨진 파일의 정보까지 모두 출력된 것을 볼 수 있다. 또한 파일의 종류에 따라 다른 색깔인 것을 볼 수 있다.

- pwd

```
kw2021202058@ubuntu:~$ pwd
/home/kw2021202058
```

pwd 명령어는 현재 작성하고 있는 파일이 위치한 경로를 출력하여 보여준다.

- cd

```
kw2021202058@ubuntu: ~/work
kw2021202058@ubuntu:~$ pwd
/home/kw2021202058
kw2021202058@ubuntu:~$ ls
2021202058 Desktop hello Music srv work
Assignment1 Documents kw Pictures srv.c
cli Downloads kw_hello.c Public Templates
cli.c examples.desktop Makefile splab_commands Videos
kw2021202058@ubuntu:~$ cd work
kw2021202058@ubuntu:~/work$ pwd
/home/kw2021202058/work
kw2021202058@ubuntu:~/work$ cd .
kw2021202058@ubuntu:~/work$ cd ..
kw2021202058@ubuntu:~$ cd work
kw2021202058@ubuntu:~/work$ cd ~
kw2021202058@ubuntu:~$ cd -
/home/kw2021202058/work
kw2021202058@ubuntu:~/work$
```

cd 명령어는 현재 directory 의 위치를 바꾸어 다른 directory 로 접근한다. cd .은 현재 directory 로 이동하는 것을 의미하고 cd ..은 현재 directory 의 상위 directory 로 이동하게 한다. cd~는 home directory 로, -는 이전 directory 로 이동하게 한다. 위 사진을 보면 pwd 명령어를 통해 현재 위치를 확인하고 ls 명령어를 사용하여 이동할 파일을 확인한 후 cd 명령어를 통해 work 로 이동하였다.

- cat

```
kw2021202058@ubuntu:~/work$ cat file1.txt
Hello This is file 1
kw2021202058@ubuntu:~/work$ cat file2.txt
Hello This is file 2
kw2021202058@ubuntu:~/work$ cat file1.txt file2.txt
Hello This is file 1
Hello This is file 2
```

cat 명령어는 파일을 연결하고 파일의 내용을 출력한다. 위 사진에서 볼 수 있듯이 cat 명령어를 사용하여 file1.txt, file2.txt 파일에 내용을 출력해주었다.

- chmod

```

kw2021202058@ubuntu:~/work$ ls -al
total 28
drwxrwxr-x 3 kw2021202058 kw2021202058 4096 Mar 19 23:12 .
drwxr-xr-x 17 kw2021202058 kw2021202058 4096 Mar 19 23:25 ..
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-r--r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ chmod u-w,g-w,o-r hello.txt
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-r--r--r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ chmod 644 hello.txt
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ chmod 664 hello.txt
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$

```

chmod 명령어는 (user, group, other, all) 대상의 파일에 대한 읽기, 쓰기, 실행 접근 권한을 변경한다. (+, -, -)연산을 통해 접근 권한을 추가, 제거, 할당을 통해 변경할 수 있다. 위 사진에서 예시를 하나 들면 chmod u-w hello.txt 를 통해 user 에게 write 권한을 제거하여 결과가 -r--이 된 것을 볼 수 있다. 또한 r(read), w(write), x(execution)를 000~777 까지 권한을 변경할 수 있다.

- mkdir

```

kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ mkdir SP_lecture
kw2021202058@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:46 SP_lecture
kw2021202058@ubuntu:~/work$

```


mkdir 명령어는 새로운 directory 를 생성한다. 위 사진에서 볼 수 있듯이 mkdir 명령어를 통해 SP_lecture 라는 새로운 directory 를 생성하였다.

- rmdir

```
kw2021202058@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058   41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:46 SP_lecture
kw2021202058@ubuntu:~/work$ rmdir SP_lecture/
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058   41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$
```

rmdir 명령어는 빈 directory 를 삭제한다. 위 사진에서 볼 수 있듯이 rmdir 명령어를 통해 SP_lecture directory 를 삭제하였다.

- rm

```
kw2021202058@ubuntu:~/work$ touch fileA.txt
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058   0 Mar 19 23:49 fileA.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058   41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ rm fileA.txt
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058  21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058   41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$
```

```

kw2021202058@ubuntu:~/work$ mkdir LINUX
kw2021202058@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:50 LINUX
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ rm -r LINUX
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$

```

```

kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file1.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ rm -i *
rm: remove regular file 'file1.txt'? y
rm: remove regular file 'file2.txt'? n
rm: remove regular file 'file3.txt'? n
rm: remove regular file 'hello.txt'? n
rm: cannot remove 'SP_lab': Is a directory
kw2021202058@ubuntu:~/work$ ls -l
total 16
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$

```

rm 명령어는 파일이나 directory를 제거한다. 첫 번째 사진을 보면 fileA.txt 파일이 제거된 것을 볼 수 있다. rm -r은 directory의 내용을 재귀적으로 제거하는 것을 말한다. 두 번째 사진에서 LINUX directory가 제거된 것을 볼 수 있다. rm -i는 세 번째 사진에서 볼 수 있듯이 제거할 파일을 물어본다.

- cp

```

kw2021202058@ubuntu:~/work$ ls -l
total 16
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ cp hello.txt hello_copy.txt
kw2021202058@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:57 hello_copy.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ cp SP_lab/* .
kw2021202058@ubuntu:~/work$ ls -l
total 28
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 15 Mar 19 23:58 fileA.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 15 Mar 19 23:58 fileC.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:57 hello_copy.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$

```

cp 명령어는 파일이나 directory 의 내용과 목록을 복사하여 새 파일을 생성한다.
 사진에서 볼 수 있듯이 hello_copy.txt 파일이 생성된 것을 볼 수 있다. cp SP_lab/* .을
 통해 SP_lab 에 있는 모든 파일을 현재 경로에 복사한 것을 볼 수 있다.

- mv

```

kw2021202058@ubuntu:~/work$ mkdir ex
kw2021202058@ubuntu:~/work$ ls
ex      file3.txt  fileC.txt  hello.txt
file2.txt fileA.txt  hello_copy.txt SP_lab
kw2021202058@ubuntu:~/work$ mv hello_copy.txt ex
kw2021202058@ubuntu:~/work$ ls
ex file2.txt file3.txt fileA.txt fileC.txt hello.txt SP_lab
kw2021202058@ubuntu:~/work$ cd ex
kw2021202058@ubuntu:~/work/ex$ ls
hello_copy.txt
kw2021202058@ubuntu:~/work/ex$ cd ..
kw2021202058@ubuntu:~/work$ mkdir LINUX
kw2021202058@ubuntu:~/work$ ls
ex file2.txt file3.txt fileA.txt fileC.txt hello.txt LINUX SP_lab
kw2021202058@ubuntu:~/work$ mv ex LINUX
kw2021202058@ubuntu:~/work$ ls
file2.txt file3.txt fileA.txt fileC.txt hello.txt LINUX SP_lab
kw2021202058@ubuntu:~/work$

```

mv 명령어는 파일이나 directory 를 이동하거나 이름을 바꾼다. 위 사진을 보면 ex 폴더가
 없어서 mkdir 명령어를 통해 만들어준 후 hello_copy.txt 파일을 ex 폴더로 이동시켰다.
 LINUX 파일을 mkdir 명령어를 통해 만들어준 후 ex 를 LINUX 폴더로 이동시켰다.

- ln

```

kw2021202058@ubuntu:~/work$ ls
file2.txt file3.txt fileA.txt fileC.txt hello.txt LINUX SP_lab
kw2021202058@ubuntu:~/work$ cat fileA.txt
This is file A
kw2021202058@ubuntu:~/work$ ln fileA.txt fileB.txt
kw2021202058@ubuntu:~/work$ cat fileB.txt
This is file A
kw2021202058@ubuntu:~/work$ vi fileB.txt
kw2021202058@ubuntu:~/work$ cat fileA.txt
This is file B
kw2021202058@ubuntu:~/work$ cat fileB.txt
This is file B
kw2021202058@ubuntu:~/work$ █

```

```

kw2021202058@ubuntu: ~/work
kw2021202058@ubuntu:~/work$ cat fileC.txt
This is file C
kw2021202058@ubuntu:~/work$ cp fileC.txt fileD.txt
kw2021202058@ubuntu:~/work$ cat fileD.txt
This is file C
kw2021202058@ubuntu:~/work$ vi fileD.txt
kw2021202058@ubuntu:~/work$ cat fileC.txt
This is file C
kw2021202058@ubuntu:~/work$ cat fileD.txt
This is file D
kw2021202058@ubuntu:~/work$

```

```

kw2021202058@ubuntu:~/work$ cat fileC.txt
This is file C
kw2021202058@ubuntu:~/work$ ln -s fileC.txt fileE.txt
kw2021202058@ubuntu:~/work$ ls -l
total 32
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 2 kw2021202058 kw2021202058 15 Mar 20 00:28 fileA.txt
-rw-rw-r-- 2 kw2021202058 kw2021202058 15 Mar 20 00:28 fileB.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 15 Mar 19 23:58 fileC.txt
lrwxrwxrwx 1 kw2021202058 kw2021202058 9 Mar 20 00:29 fileE.txt -> fileC.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 3 kw2021202058 kw2021202058 4096 Mar 20 00:11 LINUX
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ cat fileE.txt
This is file C
kw2021202058@ubuntu:~/work$ rm fileC.txt
kw2021202058@ubuntu:~/work$ cat fileE.txt
cat: fileE.txt: No such file or directory
kw2021202058@ubuntu:~/work$ █

```

ln 명령어는 파일 간에 연결을 생성한다. 연결이 생긴 파일은 같은 내용을 공유하게 된다. 우선 첫 번째 사진을 보면 cat 명령어를 사용하여 fileA.txt 에 저장되어 있는 내용을 확인했다. ln 명령어를 사용하여 fileB.txt 파일에도 같은 내용을 저장하게끔 했다. vi

editor 를 통해 fileB.txt 파일 내용을 변경한 후 cat 명령어를 통해 파일 내용을 출력한 결과 This is file B 라고 출력된 것을 확인할 수 있다. 첫 번째 사진은 hard link 의 예시이고 두 번째 사진은 cp 의 예시이다. 두 사진의 차이점은 우선 ln 은 같은 내용을 담은 이름만 다른 파일을 생성하기 때문에 내용이 수정되지만 cp 는 다른 파일을 생성하는 것으로, 파일을 수정하여도 원본 파일의 내용이 바뀌지 않는다. 세 번째 사진은 symbolic link 의 예시이며, 옵션 - s 와 함께 사용하면 symbolic link 가 생성된다. 사진에서 볼 수 있듯이 ln -s 를 통해 fileE.txt 파일이 fileC.txt 파일을 가리키게 한 것을 알 수 있다.

- touch

```
kw2021202058@ubuntu: ~/work
File Edit View Search Terminal Help
kw2021202058@ubuntu:~/work$ ls
file2.txt file3.txt fileA.txt fileB.txt fileE.txt hello.txt LINUX SP_lab
kw2021202058@ubuntu:~/work$ touch empty.txt
kw2021202058@ubuntu:~/work$ ls
empty.txt file3.txt fileB.txt hello.txt SP_lab
file2.txt fileA.txt fileE.txt LINUX
kw2021202058@ubuntu:~/work$ ls -l
total 28
-rw-rw-r-- 1 kw2021202058 kw2021202058 0 Mar 20 00:35 empty.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 2 kw2021202058 kw2021202058 15 Mar 20 00:28 fileA.txt
-rw-rw-r-- 2 kw2021202058 kw2021202058 15 Mar 20 00:28 fileB.txt
lrwxrwxrwx 1 kw2021202058 kw2021202058 9 Mar 20 00:29 fileE.txt -> fileC.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 3 kw2021202058 kw2021202058 4096 Mar 20 00:11 LINUX
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ touch empty.txt
kw2021202058@ubuntu:~/work$ ls -l
total 28
-rw-rw-r-- 1 kw2021202058 kw2021202058 0 Mar 20 00:36 empty.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 21 Mar 19 23:12 file2.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r-- 2 kw2021202058 kw2021202058 15 Mar 20 00:28 fileA.txt
-rw-rw-r-- 2 kw2021202058 kw2021202058 15 Mar 20 00:28 fileB.txt
lrwxrwxrwx 1 kw2021202058 kw2021202058 9 Mar 20 00:29 fileE.txt -> fileC.txt
-rw-rw-r-- 1 kw2021202058 kw2021202058 41 Mar 19 23:12 hello.txt
drwxrwxr-x 3 kw2021202058 kw2021202058 4096 Mar 20 00:11 LINUX
drwxrwxr-x 2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$
```

touch 명령어는 새로운 빈 파일을 생성하거나 filestamp 를 수정한다. 위 사진에서 볼 수 있듯이 empty.txt 라는 빈 파일이 생성되었고 우분투 내의 시간에 의해 00:35 -> 00:36 으로 변경된 것을 확인했다.

- ps

```

kw2021202058@ubuntu:~/work$ ps
  PID TTY          TIME CMD
 2064 pts/6        00:00:00 bash
 2456 pts/6        00:00:00 ps
kw2021202058@ubuntu:~/work$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1        0  0 00:20 ?        00:00:04 /sbin/init auto noprompt
root           2        0  0 00:20 ?        00:00:00 [kthreadd]
root           4        2  0 00:20 ?        00:00:00 [kworker/0:0H]
root           6        2  0 00:20 ?        00:00:00 [mm_percpu_wq]
root           7        2  0 00:20 ?        00:00:00 [ksoftirqd/0]
root           8        2  0 00:20 ?        00:00:00 [rcu_sched]
root           9        2  0 00:20 ?        00:00:00 [rcu_bh]
root          10        2  0 00:20 ?        00:00:00 [migration/0]
root          11        2  0 00:20 ?        00:00:00 [watchdog/0]
root          12        2  0 00:20 ?        00:00:00 [cpuhp/0]
root          13        2  0 00:20 ?        00:00:00 [cpuhp/1]
root          14        2  0 00:20 ?        00:00:00 [watchdog/1]
root          15        2  0 00:20 ?        00:00:00 [migration/1]
root          16        2  0 00:20 ?        00:00:00 [ksoftirqd/1]
root          18        2  0 00:20 ?        00:00:00 [kworker/1:0H]
root          19        2  0 00:20 ?        00:00:00 [kdevtmpfs]
root          20        2  0 00:20 ?        00:00:00 [netns]
root          21        2  0 00:20 ?        00:00:00 [rcu_tasks_kthre]
root          22        2  0 00:20 ?        00:00:00 [kauditd]
root          23        2  0 00:20 ?        00:00:00 [kworker/0:1]
root          24        2  0 00:20 ?        00:00:00 [khungtaskd]
root          25        2  0 00:20 ?        00:00:00 [oom_reaper]
root          26        2  0 00:20 ?        00:00:00 [writeback]
root          27        2  0 00:20 ?        00:00:00 [kcompactd0]
root          28        2  0 00:20 ?        00:00:00 [ksmd]
root          29        2  0 00:20 ?        00:00:00 [khugepaged]
root          30        2  0 00:20 ?        00:00:00 [crypto]
root          31        2  0 00:20 ?        00:00:00 [kintegrityd]
root          32        2  0 00:20 ?        00:00:00 [kblockd]
root          33        2  0 00:20 ?        00:00:00 [ata_sff]
root          34        2  0 00:20 ?        00:00:00 [md]
root          35        2  0 00:20 ?        00:00:00 [edac-poller]
root          36        2  0 00:20 ?        00:00:00 [devfreq_wq]
root          37        2  0 00:20 ?        00:00:00 [watchdogd]
root          41        2  0 00:20 ?        00:00:00 [kswapd0]
root          42        2  0 00:20 ?        00:00:00 [ecryptfs-kthrea]
root          84        2  0 00:20 ?        00:00:00 [kthrotld]
root          85        2  0 00:20 ?        00:00:00 [acpi_thermal_pm]
root          86        2  0 00:20 ?        00:00:00 [scsi_eh_0]
root          87        2  0 00:20 ?        00:00:00 [scsi_tmf_0]
root          88        2  0 00:20 ?        00:00:00 [scsi_eh_1]
root          89        2  0 00:20 ?        00:00:00 [scsi_tmf_1]
root          95        2  0 00:20 ?        00:00:00 [ipv6_addrconf]
root         104        2  0 00:20 ?        00:00:00 [kstrp]
root         121        2  0 00:20 ?        00:00:00 [charger_manager]
root         154        2  0 00:20 ?        00:00:01 [kworker/1:2]

```

ps 명령어는 실행중인 프로세스를 출력한다. -ef 옵션을 통해 모든 프로세스의 자세한 정보를 확인할 수 있다.

- exit

```

kw2021202058@ubuntu:~/work$ sudo apt-get install csh
[sudo] password for kw2021202058:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  csh
0 upgraded, 1 newly installed, 0 to remove and 492 not upgraded.
Need to get 235 kB of archives.
After this operation, 367 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 csh amd64 20110502-2.1ubuntu1 [235 kB]
Fetched 235 kB in 2s (94.1 kB/s)
Selecting previously unselected package csh.
(Reading database ... 180913 files and directories currently installed.)
Preparing to unpack .../csh_20110502-2.1ubuntu1_amd64.deb ...
Unpacking csh (20110502-2.1ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for doc-base (0.10.7) ...
Processing 1 added doc-base file...
Setting up csh (20110502-2.1ubuntu1) ...
update-alternatives: using /bin/bsd-csh to provide /bin/csh (csh) in auto mode
kw2021202058@ubuntu:~/work$ ps
  PID TTY          TIME CMD
 2064 pts/6        00:00:00 bash
 2888 pts/6        00:00:00 ps
kw2021202058@ubuntu:~/work$ csh
% ps
  PID TTY          TIME CMD
 2064 pts/6        00:00:00 bash
 2889 pts/6        00:00:00 csh
 2890 pts/6        00:00:00 ps
% ^C
%
% exit
% exit
kw2021202058@ubuntu:~/work$ ps
  PID TTY          TIME CMD
 2064 pts/6        00:00:00 bash
 2893 pts/6        00:00:00 ps
kw2021202058@ubuntu:~/work$ █

```

exit 명령어는 실행되고 있는 shell 을 종료하도록 한다. 위 사진은 c shell 을 설치하고 실행한 다음 exit 명령어를 통해 종료하는 것을 볼 수 있다.

- kill


```
kw2021202058@ubuntu: ~  
File Edit View Search Terminal Help  
kw2021202058@ubuntu:~$ ps  
  PID TTY          TIME CMD  
 2913 pts/2    00:00:00 bash  
 2973 pts/2    00:00:00 ps  
kw2021202058@ubuntu:~$ vi hello  
[1]+  Stopped                  vi hello  
kw2021202058@ubuntu:~$ ps  
  PID TTY          TIME CMD  
 2913 pts/2    00:00:00 bash  
 2974 pts/2    00:00:00 vi  
 2977 pts/2    00:00:00 ps  
kw2021202058@ubuntu:~$ kill -9 2974  
kw2021202058@ubuntu:~$ ps  
  PID TTY          TIME CMD  
 2913 pts/2    00:00:00 bash  
 2978 pts/2    00:00:00 ps  
[1]+  Killed                  vi hello  
kw2021202058@ubuntu:~$
```

kill 명령어는 process 에 신호를 보낸다. 첫 번째 사진은 yes my name 을 입력하면 my name 이 무한으로 출력되고, 다른 terminal 에서 ps -e | tail 을 통해 yes 의 process 를 찾아내 kill 과 함께 입력하면 my name 이 출력되는 것이 중단되는 것을 볼 수 있다. 두 번째 사진은 kill -9 를 통해 강제 종료를 하는 것이다.

- passwd

```
kw2021202058@ubuntu:~$ passwd  
Changing password for kw2021202058.  
(current) UNIX password:  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
kw2021202058@ubuntu:~$
```

passwd 명령어는 사용자의 비밀번호를 바꾸게 하는 것으로, 기존의 비밀번호에서 새로운 비밀번호로 변경하였다.

- uname

```
kw2021202058@ubuntu:~$ uname
Linux
kw2021202058@ubuntu:~$ uname -r
4.15.0-29-generic
kw2021202058@ubuntu:~$ uname -m
x86_64
kw2021202058@ubuntu:~$ uname -a
Linux ubuntu 4.15.0-29-generic #31~16.04.1-Ubuntu SMP Wed Jul 18 08:54:04 UTC 20
18 x86_64 x86_64 x86_64 GNU/Linux
kw2021202058@ubuntu:~$
```

uname 명령어는 시스템의 정보를 출력한다. 옵션에 따라 -r 은 kernel realease, -m 은 machine hardware 의 이름, -a 는 모든 정보를 출력한다.

- wc

```
kw2021202058@ubuntu:~$ ls
Assignment1 Desktop examples.desktop Pictures srv Videos
cli Documents makefile Public srv.c work
cli.c Downloads Music splab_commands Templates
kw2021202058@ubuntu:~$ cd work
kw2021202058@ubuntu:~/work$ cat hello.txt
hello world
My Name is N~~~
How are you?
kw2021202058@ubuntu:~/work$ wc hello.txt
 3  9 41 hello.txt
kw2021202058@ubuntu:~/work$
```

wc 명령어는 문서의 행, 단어, 문자의 개수를 계산하여 보여준다. 위 사진에서 볼 수 있듯이 hello.txt 파일의 행, 단어, 문자의 개수를 보여주는 것을 알 수 있다.

- echo

```
kw2021202058@ubuntu:~/work$ echo helloworld
helloworld
kw2021202058@ubuntu:~/work$ echo $HOME
/home/kw2021202058
kw2021202058@ubuntu:~/work$ echo ~
/home/kw2021202058
kw2021202058@ubuntu:~/work$
```

echo 명령어는 텍스트의 한 행을 보여준다. echo 명령어 뒤에 온 문자열을 출력한다.

- alias

```

kw2021202058@ubuntu:~/work$ myls
No command 'mysl' found, did you mean:
  Command 'tyls' from package 'terminology' (universe)
  Command 'mmls' from package 'sleuthkit' (universe)
mysl: command not found
kw2021202058@ubuntu:~/work$ alias mysl='ls -al'
kw2021202058@ubuntu:~/work$ mysl
total 36
drwxrwxr-x  4 kw2021202058 kw2021202058 4096 Mar 20 00:35 .
drwxr-xr-x 17 kw2021202058 kw2021202058 4096 Mar 20 00:43 ..
-rw-rw-r--  1 kw2021202058 kw2021202058   0 Mar 20 00:36 empty.txt
-rw-rw-r--  1 kw2021202058 kw2021202058  21 Mar 19 23:12 file2.txt
-rw-rw-r--  1 kw2021202058 kw2021202058 2001 Mar 19 23:12 file3.txt
-rw-rw-r--  2 kw2021202058 kw2021202058  15 Mar 20 00:28 fileA.txt
-rw-rw-r--  2 kw2021202058 kw2021202058  15 Mar 20 00:28 fileB.txt
lrwxrwxrwx  1 kw2021202058 kw2021202058   9 Mar 20 00:29 fileE.txt -> fileC.txt
-rw-rw-r--  1 kw2021202058 kw2021202058  41 Mar 19 23:12 hello.txt
drwxrwxr-x  3 kw2021202058 kw2021202058 4096 Mar 20 00:11 LINUX
drwxrwxr-x  2 kw2021202058 kw2021202058 4096 Mar 19 23:12 SP_lab
kw2021202058@ubuntu:~/work$ alias
alias alert='notify-send --urgency=low -i "${?} = 0 ] && echo terminal || echo
error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[\;&|]\s*alert$//'\`'
)'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
alias mysl='ls -al'
kw2021202058@ubuntu:~/work$

```

alias 명령어는 임의로 다른 단어로 대체하여 사용할 수 있도록 한다. 위 사진에서 볼 수 있듯이, ls-al 명령어를 mysl 로 대체하여 사용한 것을 볼 수 있다. 또한 unalias 를 통해 설정한 단어를 해제 할 수도 있다.

- grep

```

kw2021202058@ubuntu:~/work$ vi text.txt
kw2021202058@ubuntu:~/work$ cat text.txt
hello world
My Name is N~~~~
How are you?
kw2021202058@ubuntu:~/work$ grep hello text.txt
hello world
kw2021202058@ubuntu:~/work$

```

grep 명령어는 파일에서 찾고자 하는 어떤 단어를 찾아 표시한다. 위 사진에서 text.txt 파일이 없기 때문에 vi editor 를 통해 파일을 만들어준 후 cat 명령어를 통해 text.txt 파일을 출력해주었다. grep 명령어를 통해 hello 라는 단어를 찾으려 하였다.

2-3. Linux_basedProgramming

- Vi editor

작년 2 학년 2 학기 때 데이터구조설계, 실습 시간에 vmware 와 Ubuntu 를 사용해봤는데 명령어 위주 보다는, 코드를 구현하고 실행, 압축 등의 명령어만 활용하여 과제를 제출했기 때문에 명령어를 하나하나 실행해보는 이번 실습이 낯설고 헛갈렸다. 특히 rm 과 rmdir 은, 전자는 directory 와 파일 둘 다를 삭제할 수 있지만, 후자는 directory 만 제거할 수 있다. 공통점은 빈 파일일 경우에 삭제할 수 있지만, 차이점은 directory 의 제거 유무인데 rm 만 사용해도 될 거 같다는 의문점이 들기도 하고 헛갈렸다. 또한 rm -r 이 명령어는 재귀적으로 삭제해 나가는데, 정확한 의미가 이해가 가지 않아 직접 실험을 해보니 rm -r 은 파일이나 directory 가 비어 있지 않아도 삭제가 되며, 파일이나 폴더가 빌 때 까지 재귀적으로 지운다는 것을 알게 되었다. 또한 man 명령어에서 man 도 해당 명령어의 사용법을 알 수 있고, man -a 도 사용법을 알 수 있는데 두 개의 차이점이 무엇인지 궁금해 직접 실습해보니, 전자는 하위 섹션을 출력하고, 후자는 전체 섹션을 출력한다는 것을 알 수 있었다. 하나하나 비교하고 실험해가며 실습을 진행하다 보니 비록 시간이 많이 걸렸지만, 모든 것의 기본이 된다고 생각하니 의미 있었다. 또한 마우스 커서로 위치를 지정하지 못하고, 방향키를 이용해서 입력 위치를 지정해야 한다는 사실이 불편하고 어색하게 느껴졌다. 또한 vi editor 에서 'I'를 통해 insert 모드로 바꾸고 'esc' + 'shift :'를 통해 명령모드로 가서 저장할지 종료할지 설정하는 부분도 헛갈리고 낯설어 익숙해지는 데 많은 시간이 걸렸다. 이번 과제를 통해 LINUX 환경에 익숙해지는 과정을 거쳤고 더 적응하기 위해 과제 이외의 시간을 활용한 연습의 필요성을 느꼈다.

4. Reference

시스템프로그래밍실습 강의자료