시스템프로그래밍실습 보고서

Basic 과제

과 목 시스템프로그래밍실습

담당교수 이기훈교수님

학 과 컴퓨터정보공학부

학 번 2021202058

이 름 송채영

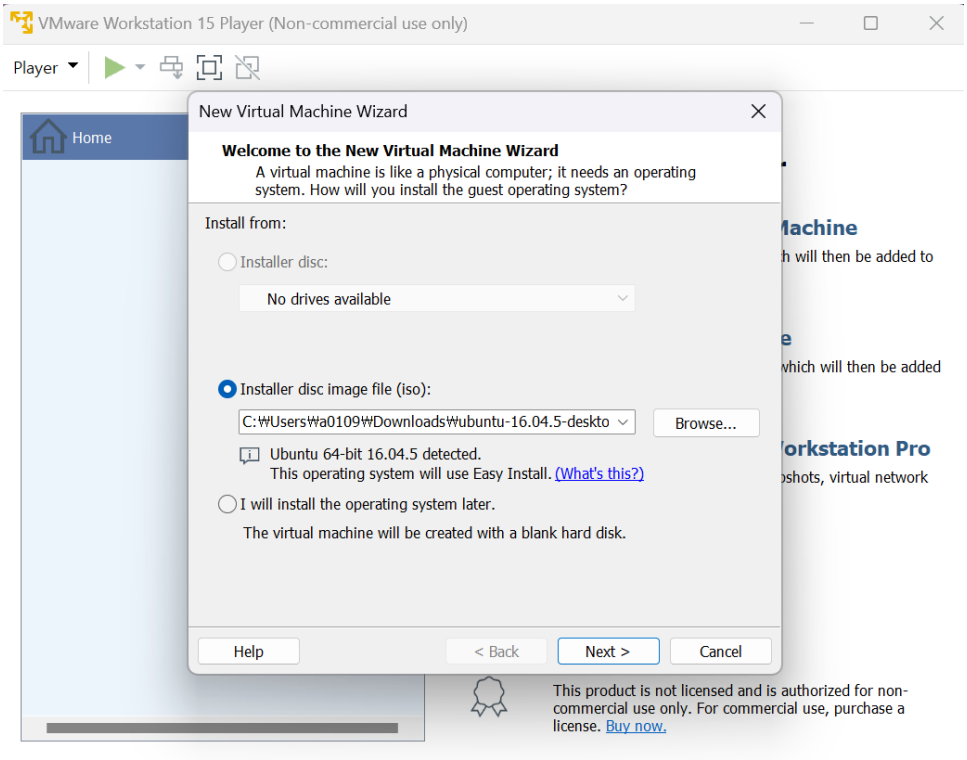
1. Introduction

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.

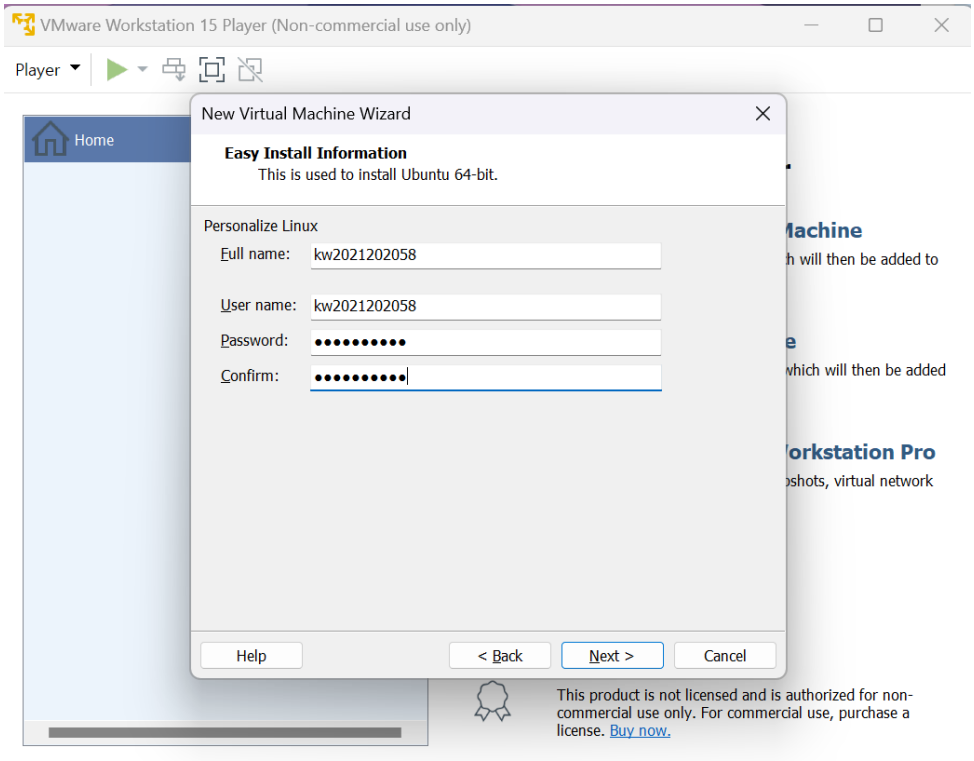
2. 결과 화면

2-1. Ubuntu Installation

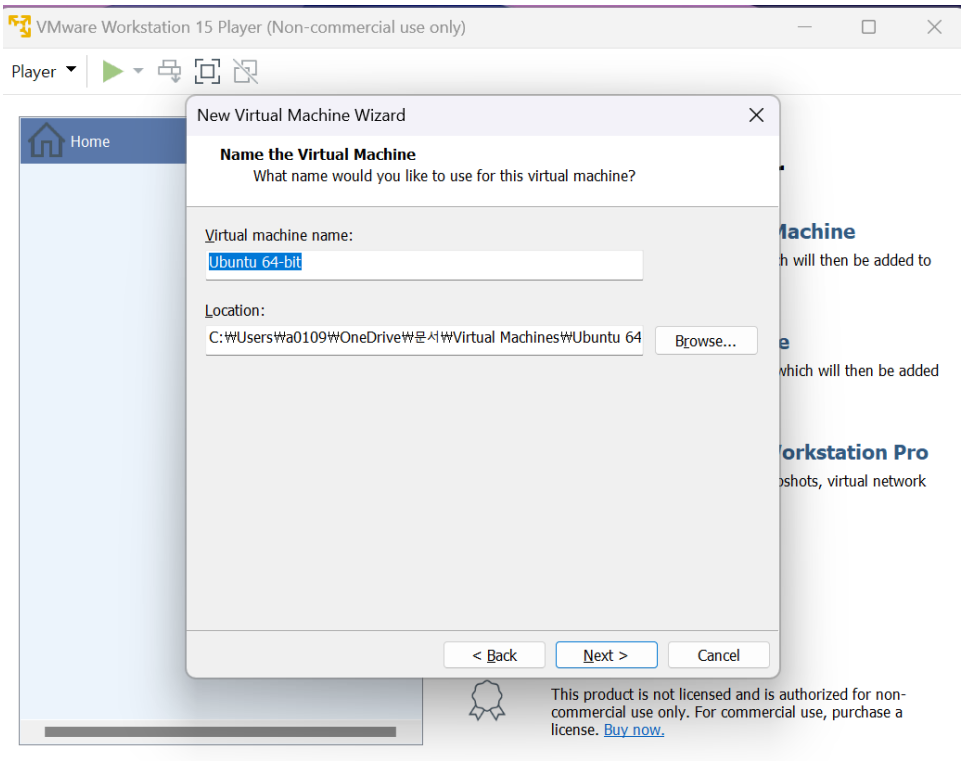
Ubuntu 16.04.5 와 Vmware 15.5.7 을 설치하였다. 옵션 선택은 기본 값을 유지하였다. 설치 과정은 과제에서 제외이므로 Vmware 의 설치과정은 생략한다.



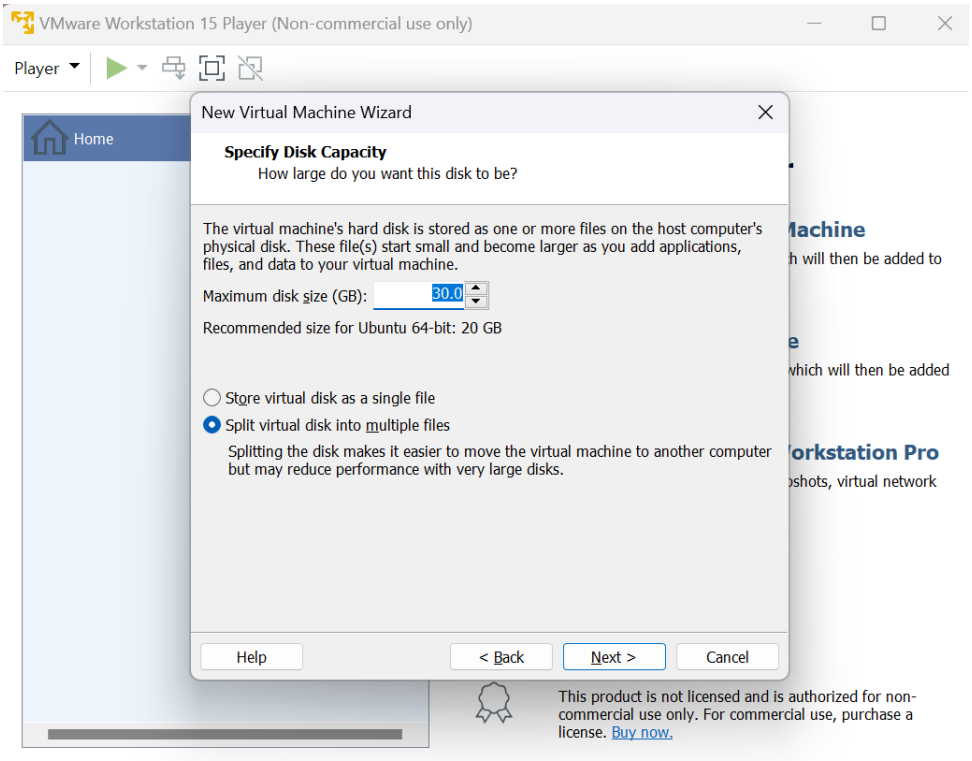
가상머신을 생성하는 과정으로, VMware 를 이용해 Ubuntu 를 설치하였다. 위의 사진은 Ubuntu ISO 파일 위치를 지정해주는 과정을 캡처한것이다.



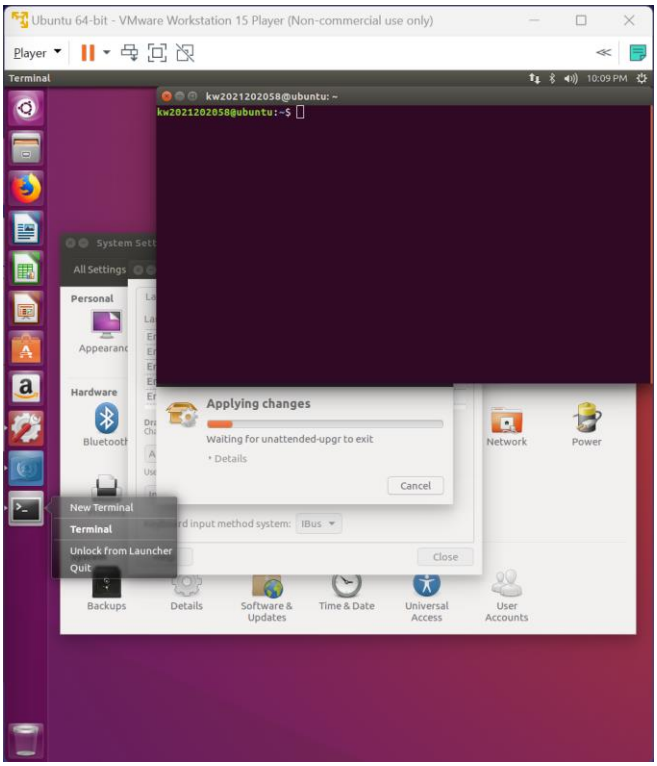
위의 사진은 Ubuntu 에서 로그인 할 계정을 설정하기 위해 User name 인 ID 와 비밀번호인 Pssword 와 Confirm 을 지정해주었다. 과제에서 계정 ID 는 kw+학번으로 지정되어 있기 때문에 kw2021202058 로 설정해주었다.



위 사진은 가상머신의 이름과 설치될 위치를 지정해주는 과정을 캡처한것이다.



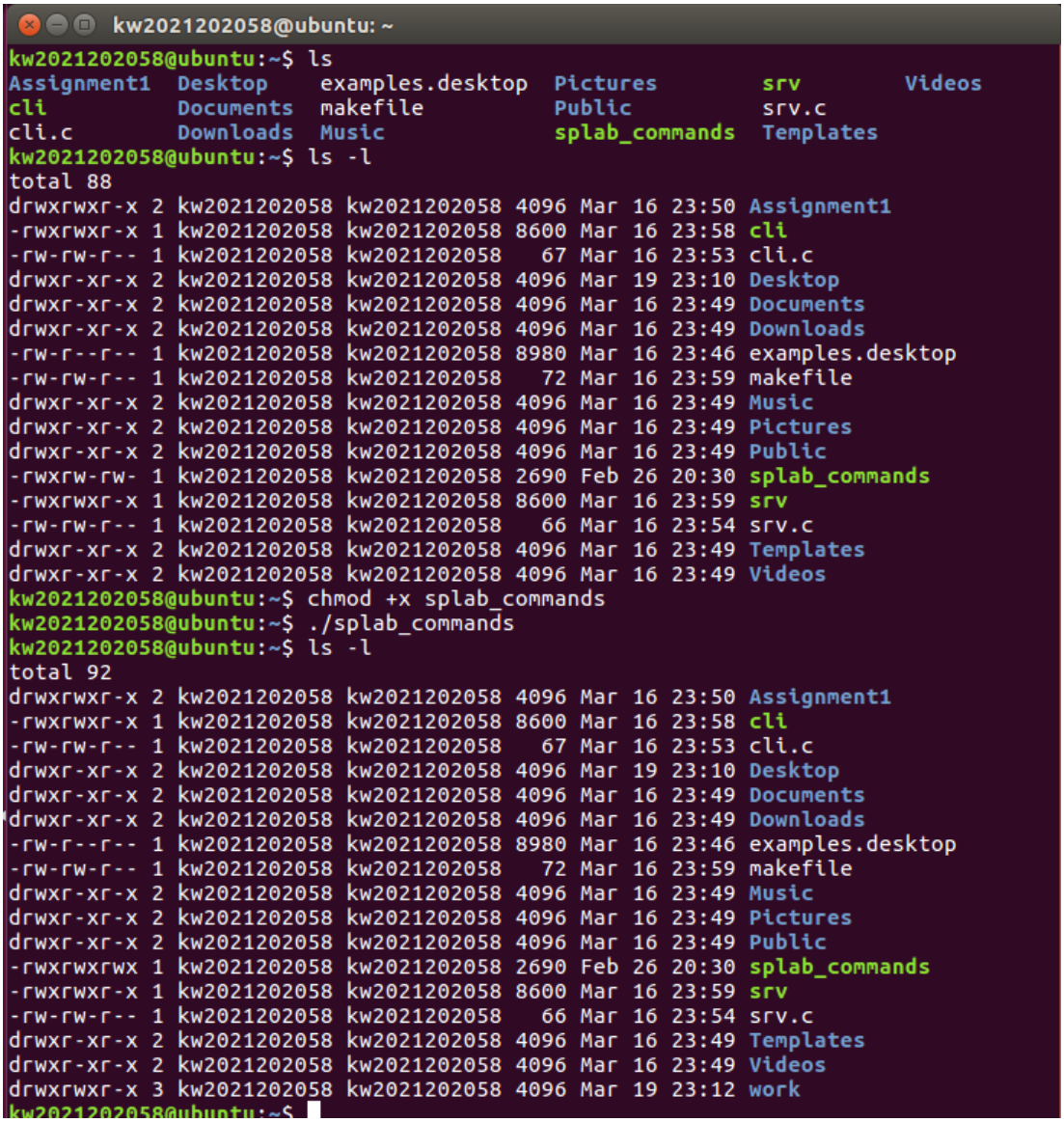
가상머신에 할당할 저장장치의 크기를 설정해주는 과정을 캡처한 사진이다. 가상머신에 할당할 메인 메모리의 크기와 cpu 그리고 cpu core 의 수를 설정해주는 과정이다. Dick Size 는 30 으로 설정해주었다.



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

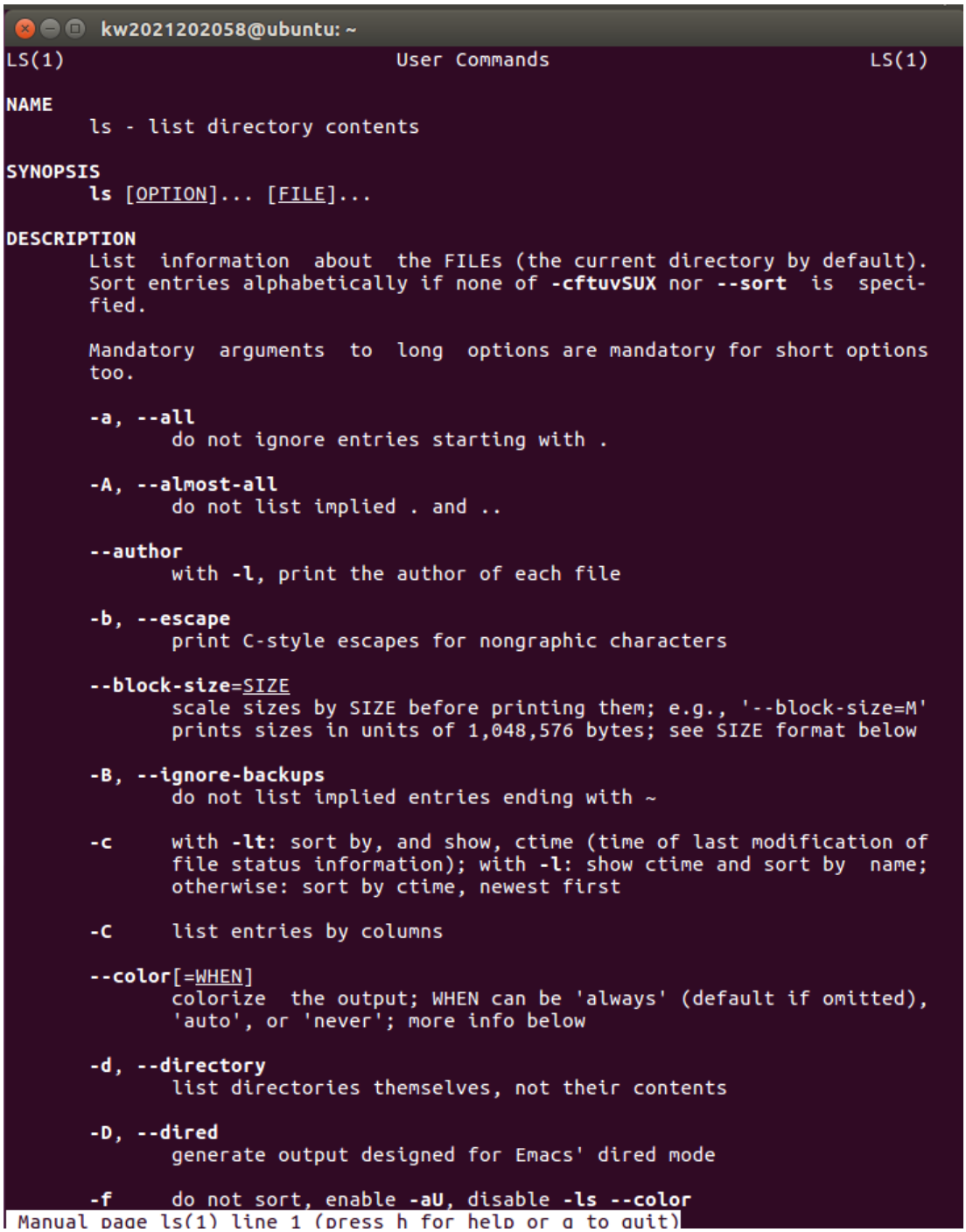


실습 자료와 같이 올려 주신 'splab_commands' 파일을 실행해주어 work 파일이 생성된 것을 확인할 수 있다.

- man

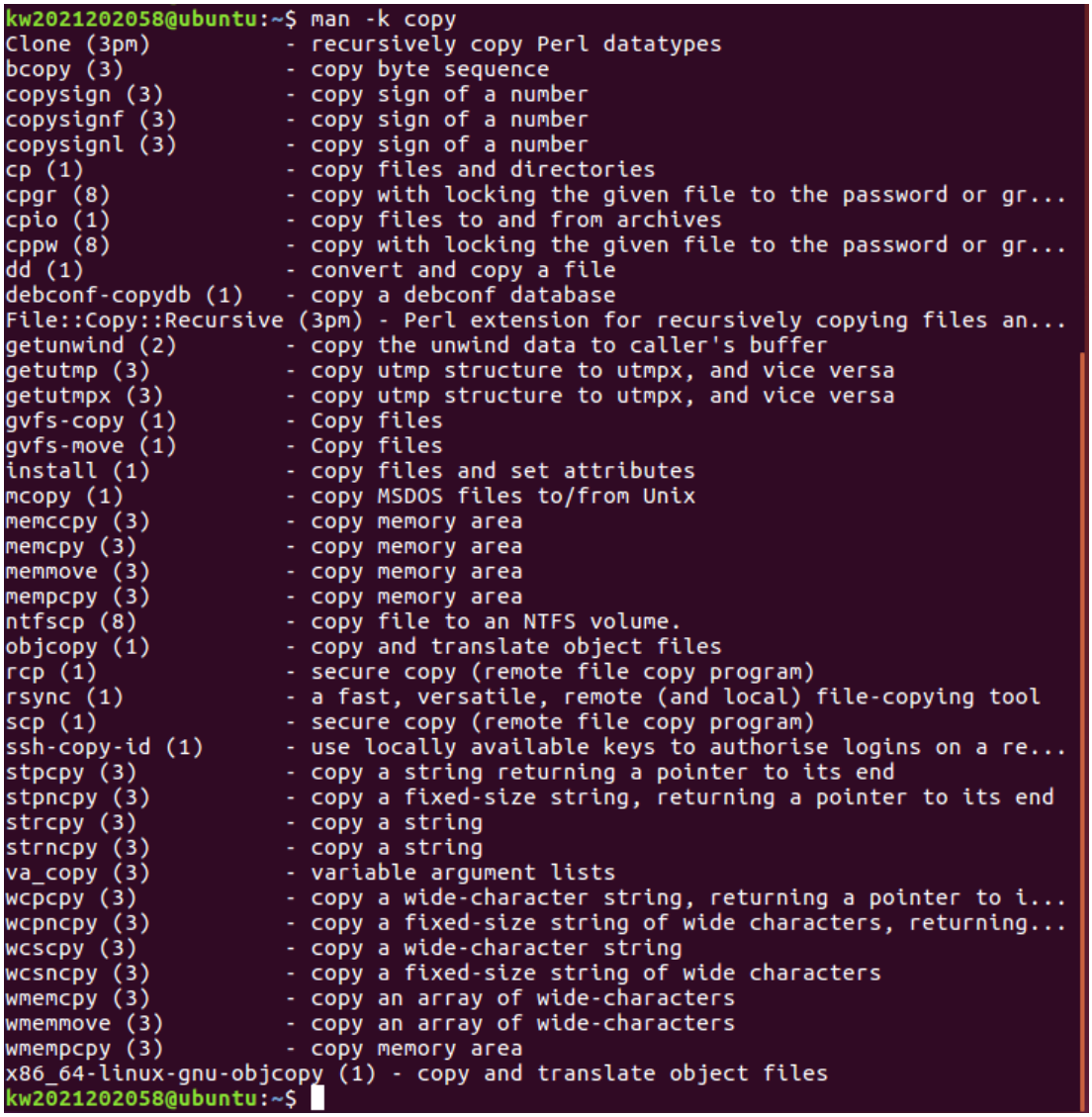
man 명령어를 사용하여 해당 명령어의 메뉴얼(이름, 사용 형식, 개요)과 프로그램의 사용법을 확인할 수 있다. man 명령어는 man 다음에 찾고 싶은 명령어를 입력하여 사용한다.

- man ls



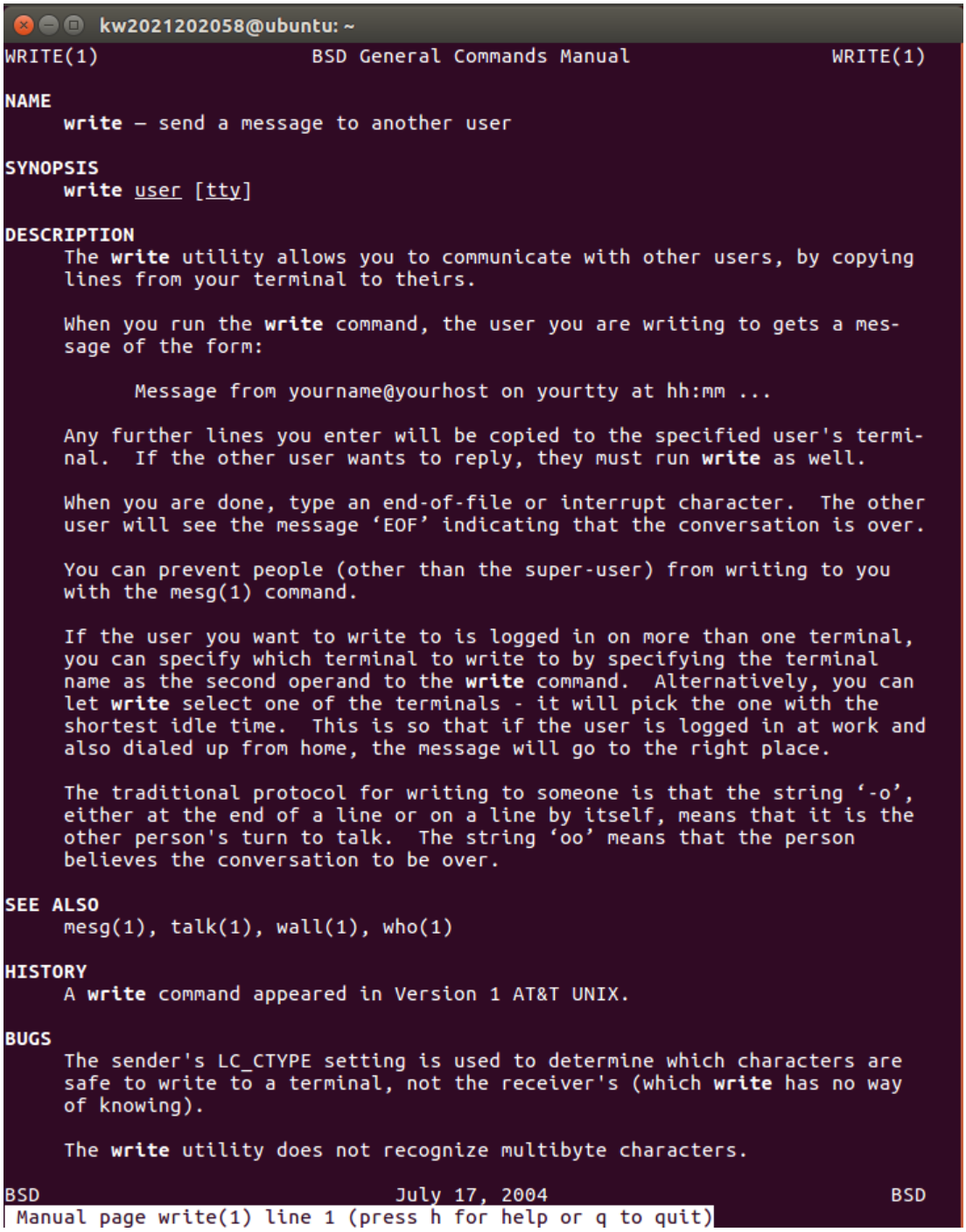
위 사진과 같이 man ls 를 입력 시 ls 명령어에 대한 매뉴얼을 보여준다.

- man -k copy



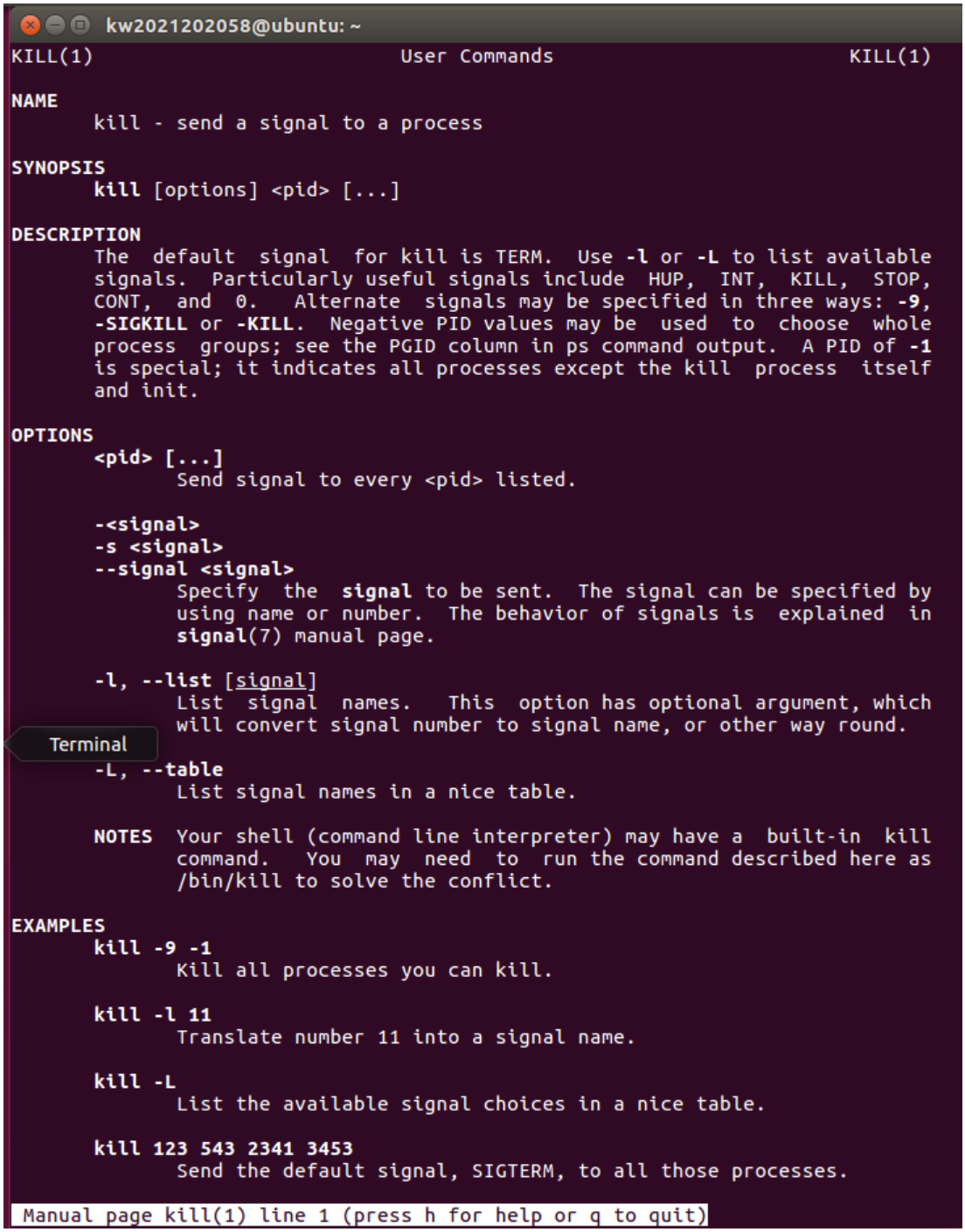
위 사진과 같이 man -k copy 를 입력 시 copy 단어가 포함된 모든 명령어를 보여준다.

- man -a write



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



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

- ls

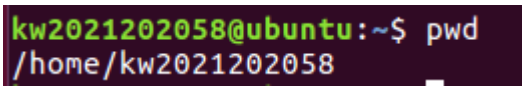
```
kw2021202058@ubuntu:~$ ls
Assignment1  Desktop    examples.desktop  Pictures        srv         Videos
cli          Documents  makefile          Public          srv.c       work
cli.c        Downloads  Music             splab_commands  Templates
kw2021202058@ubuntu:~$ ls -a
.              Downloads          srv
..             examples.desktop   srv.c
Assignment1    .gconf             .sudo_as_admin_successful
.bash_history  .gnupg             Templates
.bash_logout   .ICEauthority      Videos
.bashrc        .lesshst           .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-------  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-------  1 kw2021202058 kw2021202058 1908 Mar 19 23:08 .ICEauthority
-rw-------  1 kw2021202058 kw2021202058   28 Mar 19 23:25 .lesshst
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-------  1 kw2021202058 kw2021202058 2539 Mar 17 00:00 .viminfo
drwxrwxr-x  3 kw2021202058 kw2021202058 4096 Mar 19 23:12 work
-rw-------  1 kw2021202058 kw2021202058   51 Mar 19 23:08 .Xauthority
-rw-rw-r--  1 kw2021202058 kw2021202058  131 Mar 18 22:32 .xinputrc
-rw-------  1 kw2021202058 kw2021202058   82 Mar 19 23:08 .xsession-errors
-rw-------  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

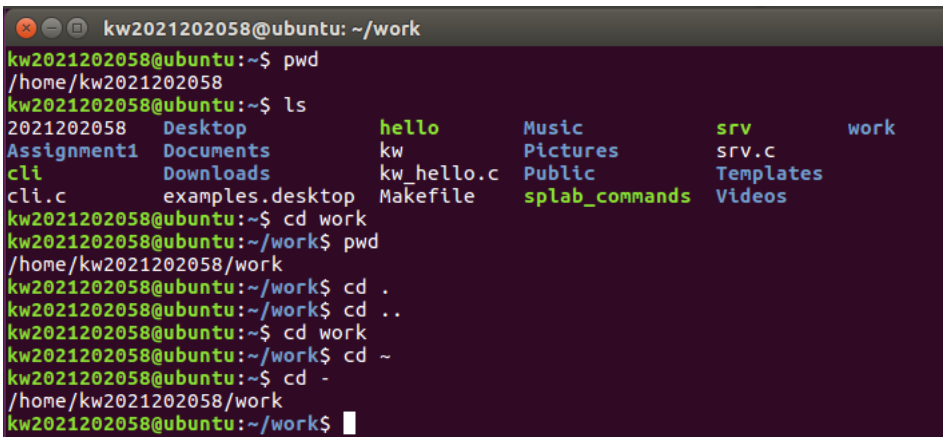
directory 를 의미하며, '*'는 실행파일을 의미한다. ls -l 를 입력 시 파일 정보를 자세하게 출력한다. 두 번째 사진에서는 ls -al 을 입력해주었는데, 숨겨진 파일의 정보까지 모두 출력된 것을 볼 수 있다. 또한 파일의 종류에 따라 다른 색깔인 것을 볼 수 있다.

- pwd



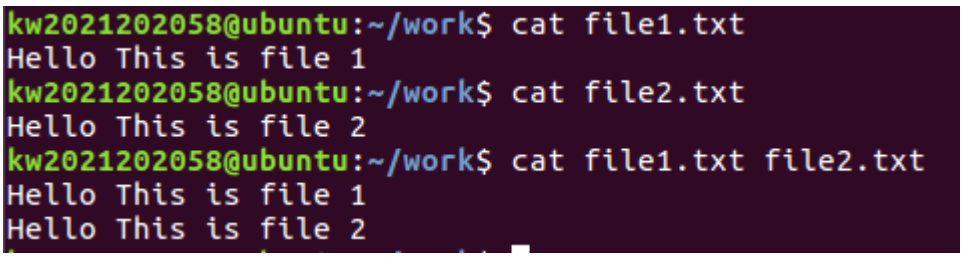
pwd 명령어는 현재 작성하고 있는 파일이 위치한 경로를 출력하여 보여준다.

- cd



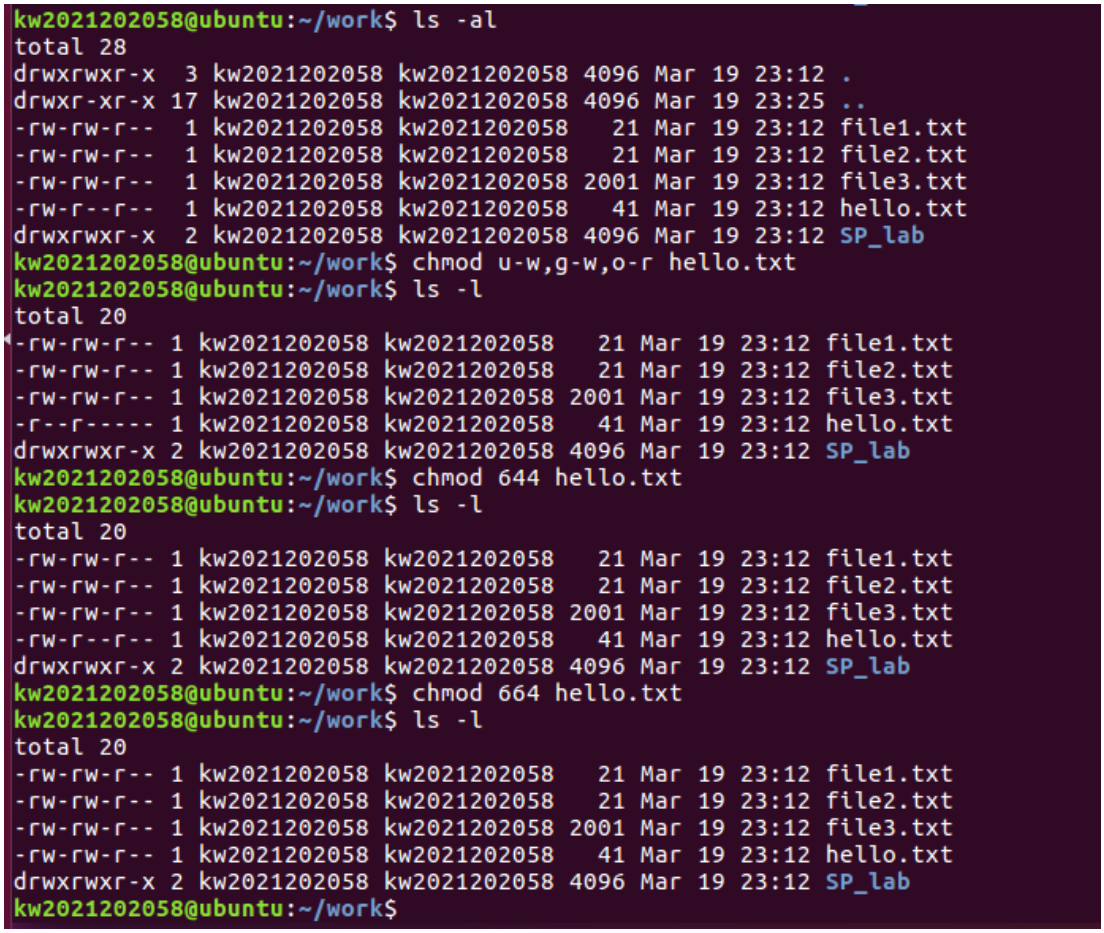
cd 명령어는 현재 directory 의 위치를 바꾸어 다른 directory 로 접근한다. cd .은 현재 directory 로 이동하는 것을 의미하고 cd . .은 현재 directory 의 상위 directory 로 이동하게 한다. cd~는 home directory 로, -는 이전 directory 로 이동하게 한다. 위 사진을 보면 pwd 명령어를 통해 현재 위치를 확인하고 ls 명령어를 사용하여 이동할 파일을 확인한 후 cd 명령어를 통해 work 로 이동하였다.

- cat



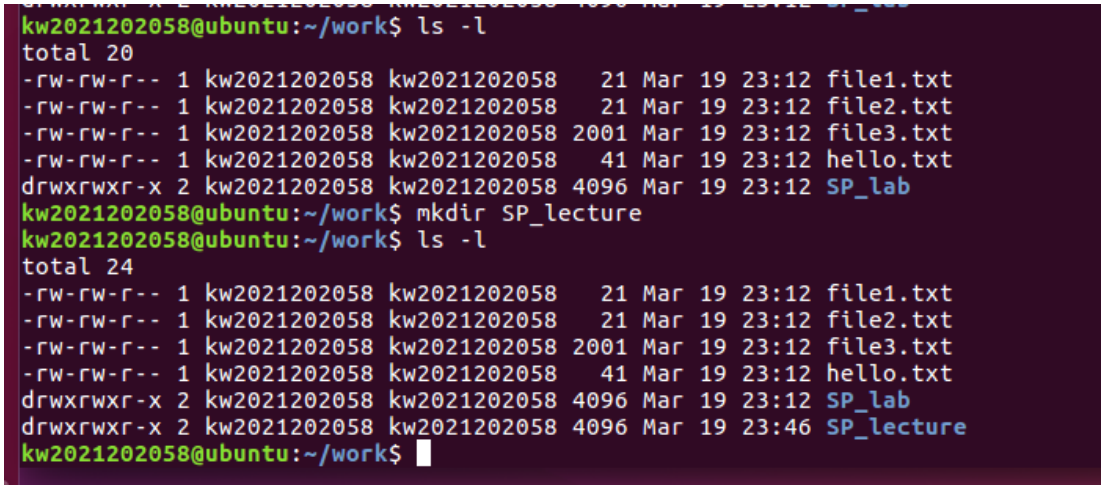
cat 명령어는 파일을 연결하고 파일의 내용을 출력한다. 위 사진에서 볼 수 있듯이 cat 명령어를 사용하여 file1.txt, file2.txt 파일에 내용을 출력해주었다.

- chmod



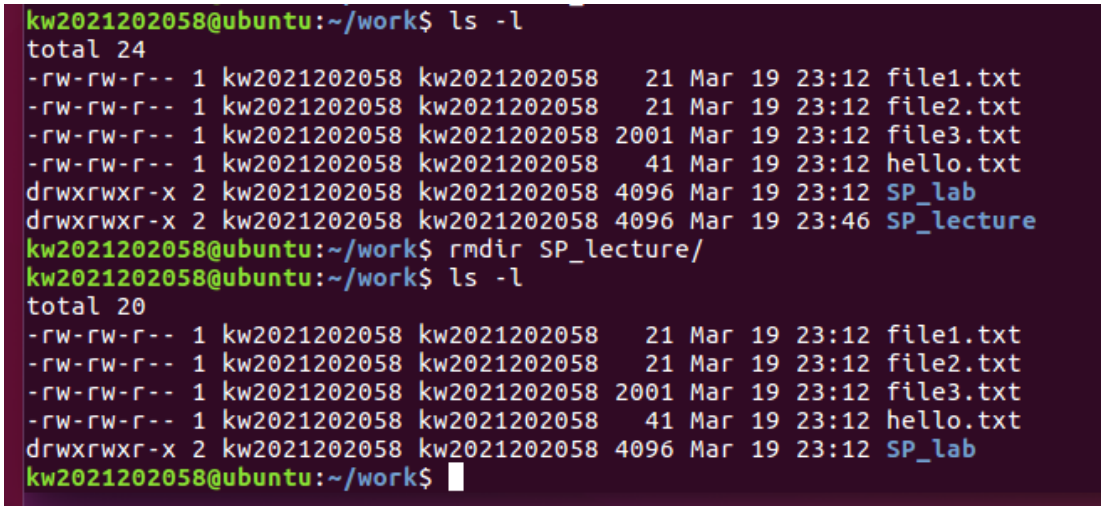
chmod 명령어는 (user, group, other, all) 대상의 파일에 대한 읽기, 쓰기, 실행 접근 권한을 변경한다. (+, -, -)연산을 통해 접근 권한을 추가, 제거, 할당을 통해 변경할 수 있다. 위 사진에서 예시를 하나 들면 chmod u-w hello.txt 를 통해 user 에게 write 권한을 제거하여 결과가 -r--이 된 것을 볼 수 있다. 또한 r(read), w(write), x(execution)를 000~777 까지 권한을 변경할 수 있다.

- mkdir



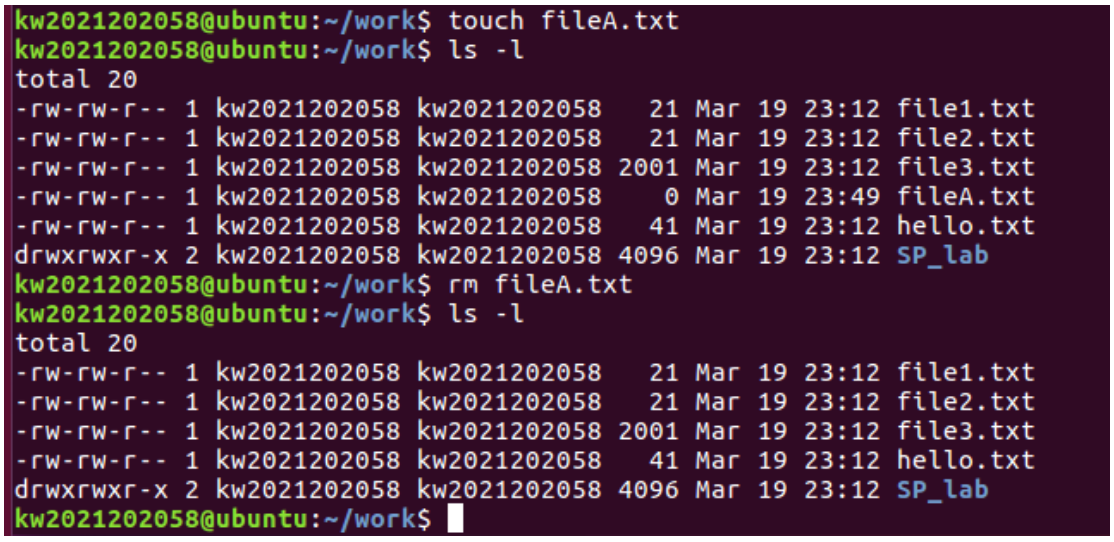
mkdir 명령어는 새로운 directory 를 생성한다. 위 사진에서 볼 수 있듯이 mkdir 명령어를 통해 SP_lecture 라는 새로운 directory 를 생성하였다.

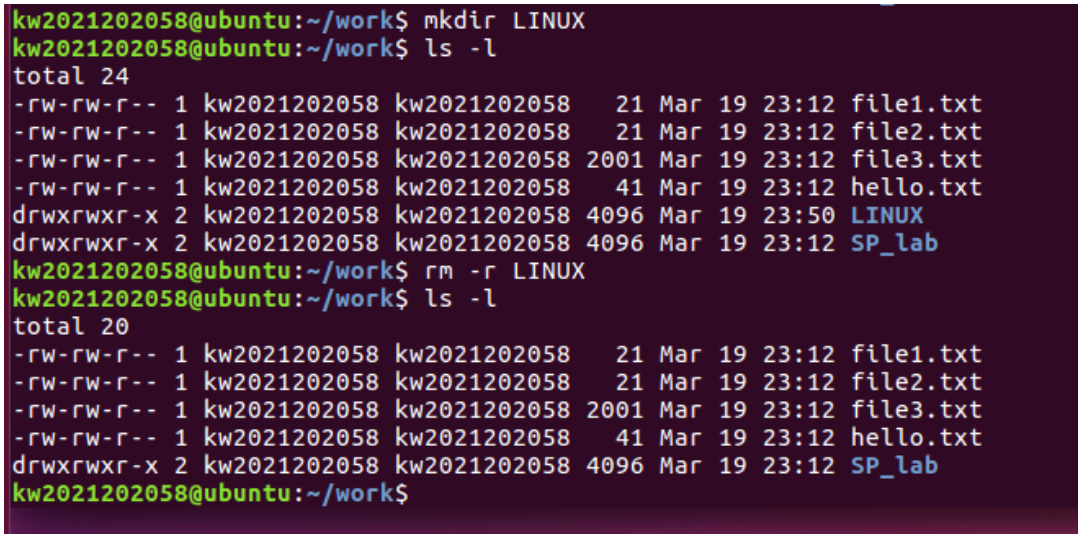
- rmdir

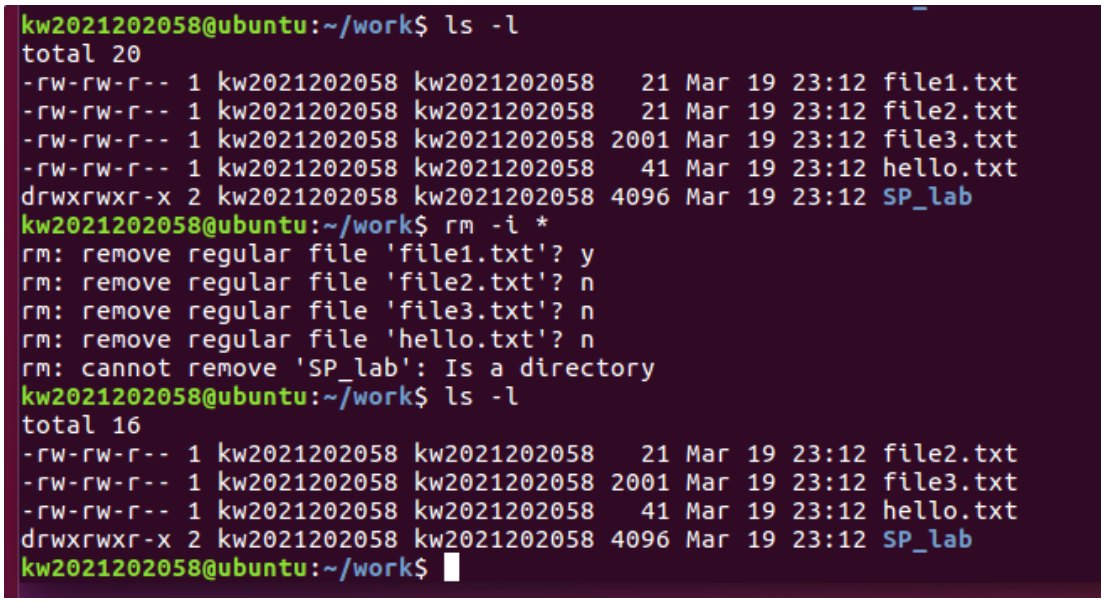


rmdir 명령어는 빈 directory 를 삭제한다. 위 사진에서 볼 수 있듯이 rmdir 명령어를 통해 SP_lecture directory 를 삭제하였다.

- rm







rm 명령어는 파일이나 directory 를 제거한다. 첫 번째 사진을 보면 fileA.txt 파일이 제거된 것을 볼 수 있다. rm -r 은 directory 의 내용을 재귀적으로 제거하는 것을 말한다. 두 번째 사진에서 LINUX directory 가 제거된 것을 볼 수 있다. rm -i 는 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$
```

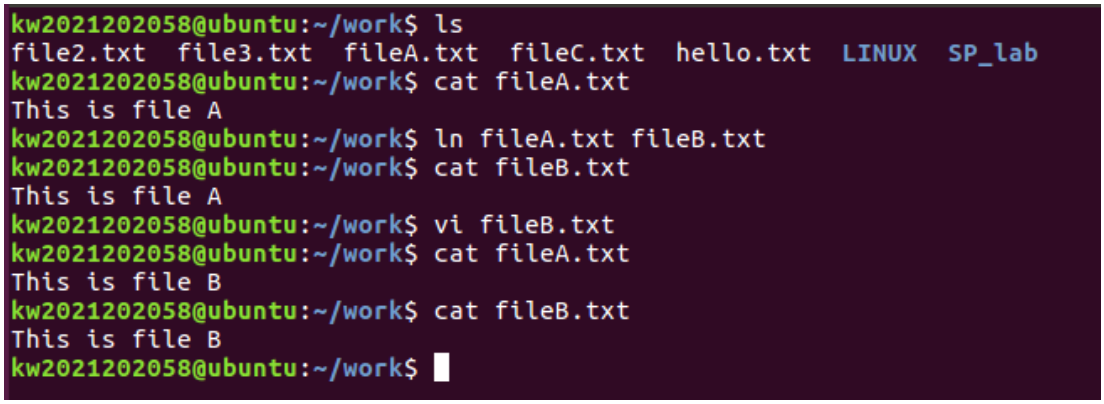
cp 명령어는 파일이나 directory 의 내용과 목록을 복사하여 새 파일을 생성한다. 사진에서 볼 수 있듯이 hello_copy.txt 파일이 생성된 것을 볼 수 있다. cp SP_lab/* .을 통해 SP_lab 에 있는 모든 파일을 현재 경로에 복사한 것을 볼 수 있다.

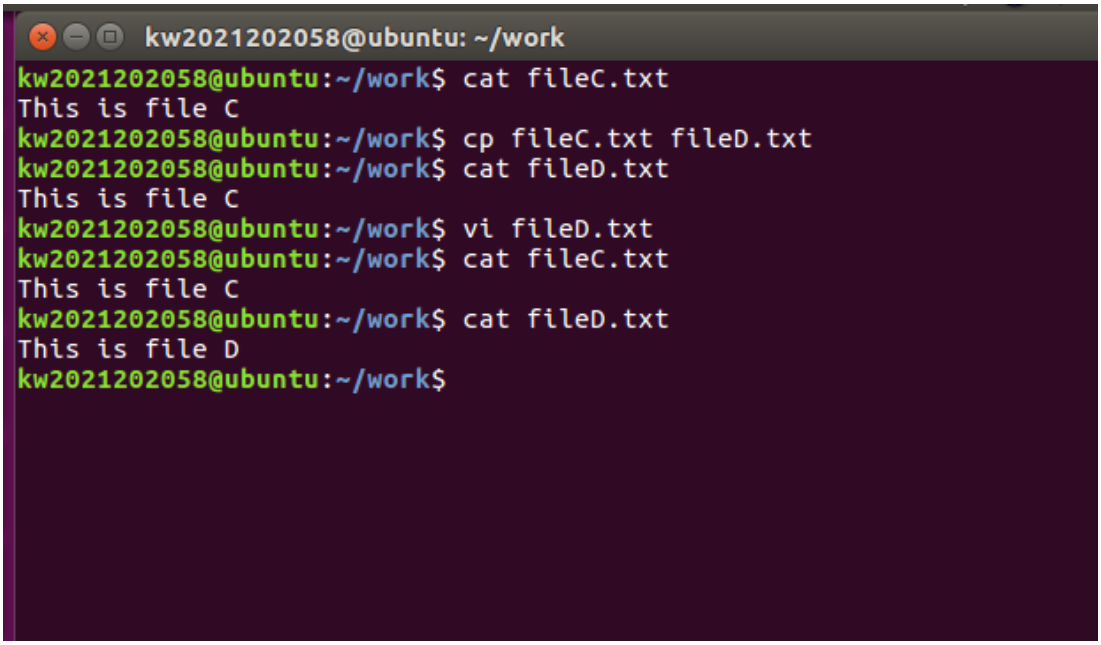
- mv

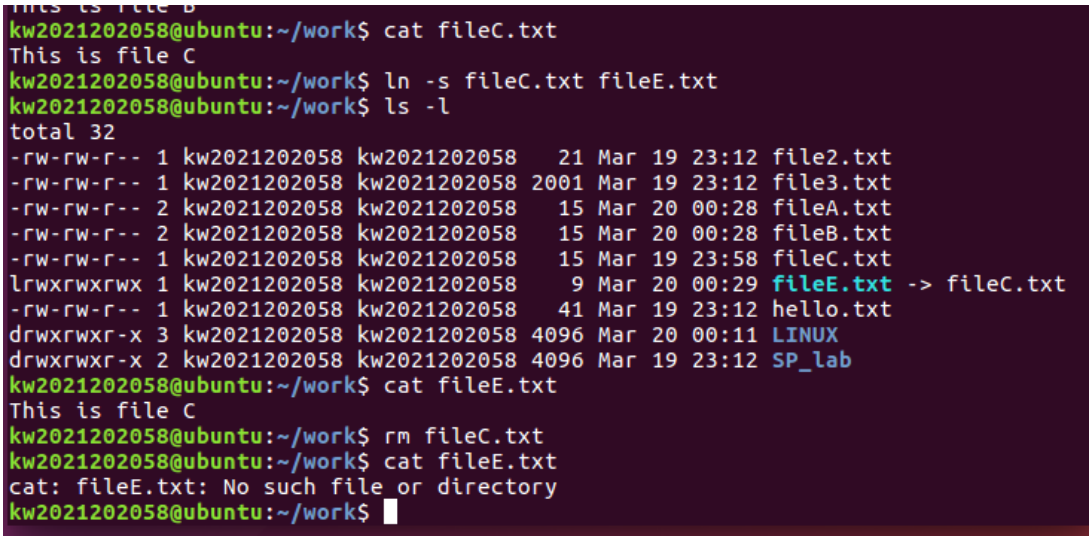
```
kw2021202058@ubuntu:~/work$ mkdir ex
kw2021202058@ubuntu:~/work$ ls
ex          file3.txt  fileC.txt       hello.txt
file2.txt   fileA.txt  hello_copy.txt  SP_lab
kw2021202058@ubuntu:~/work$ mv hello_copy.txt ex
kw2021202058@ubuntu:~/work$ ls
ex  file2.txt  file3.txt  fileA.txt  fileC.txt  hello.txt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



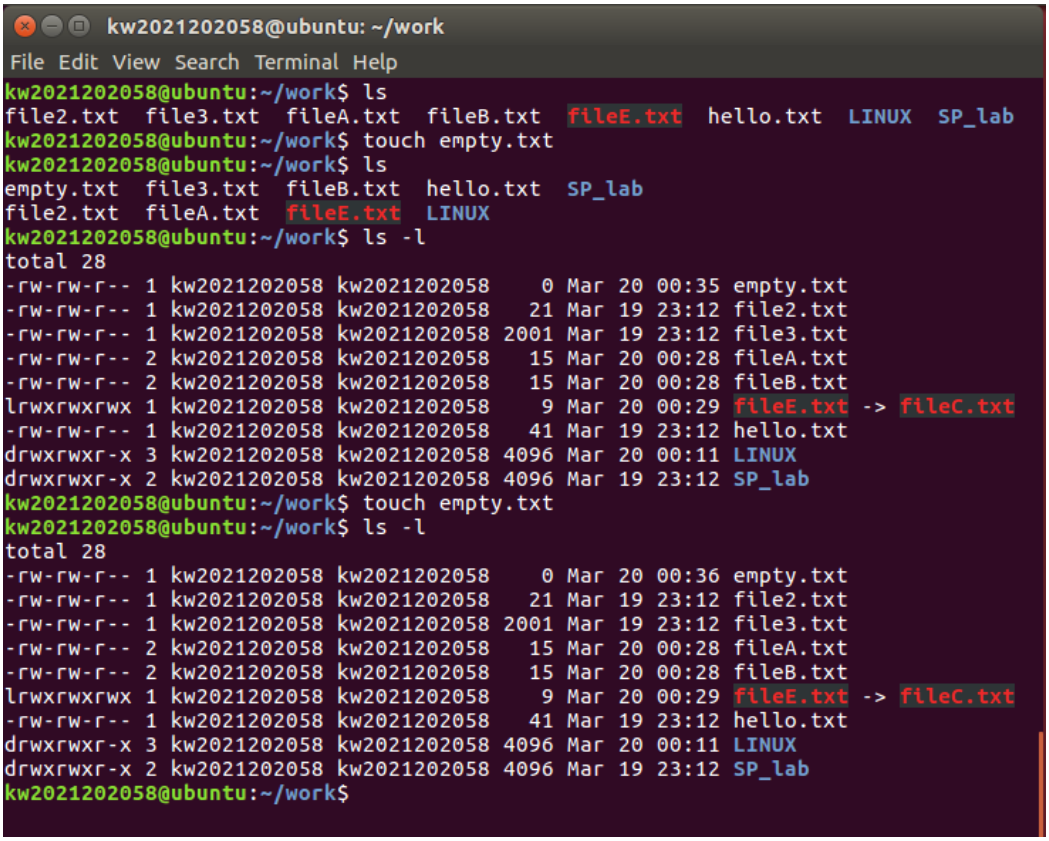




ln 명령어는 파일 간에 연결을 생성한다. 연결이 생긴 파일은 같은 내용을 공유하게 된다. 우선 첫 번째 사진을 보면 cat 명령어를 사용하여 fileA.txt 에 저장되어 있는 내용을 확인했다. ln 명령어를 사용하여 fileB.txt 파일에도 같은 내용을 저장하게끔 했다. vi

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



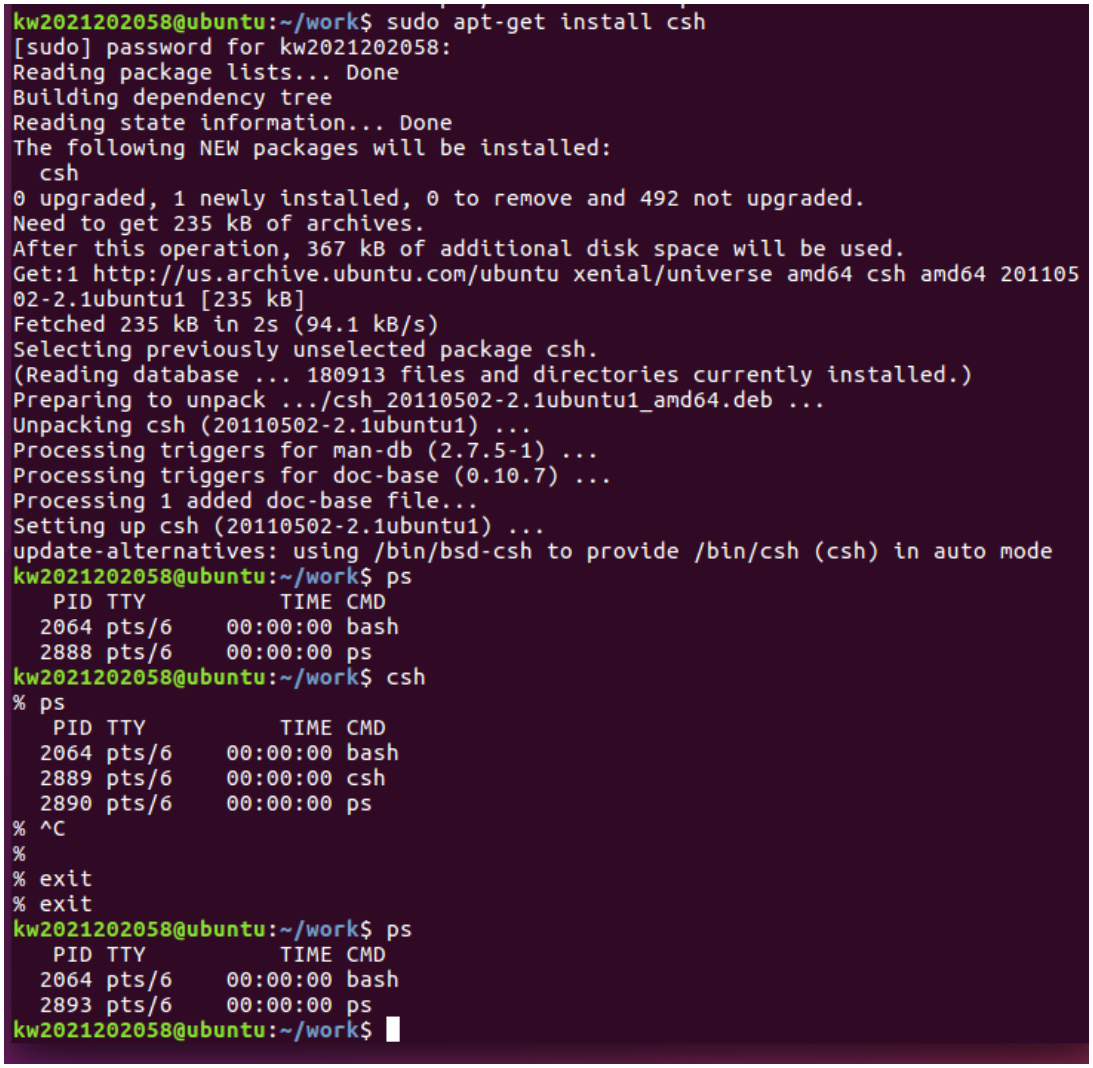
touch 명령어는 새로운 빈 파일을 생성하거나 filestamp 를 수정한다. 위 사진에서 볼 수 있듯이 empty.txt 라는 빈 파일이 생성되었고 우분투 내의 시간에 의해 00:35 -> 00:36 으로 변경된 것을 확인했다.

- 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]
```

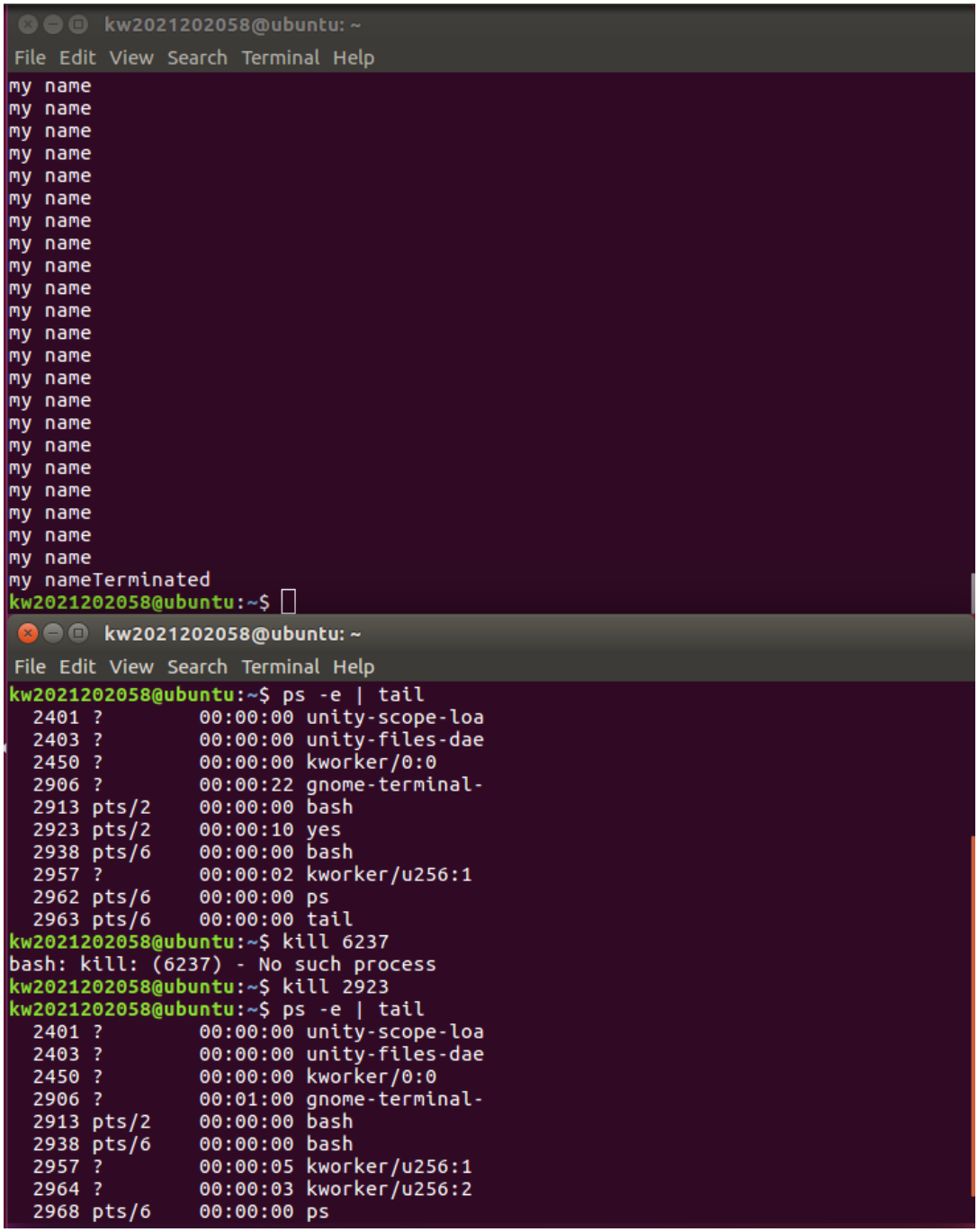
ps 명령어는 실행중인 프로세스를 출력한다. -ef 옵션을 통해 모든 프로세스의 자세한 정보를 확인할 수 있다.

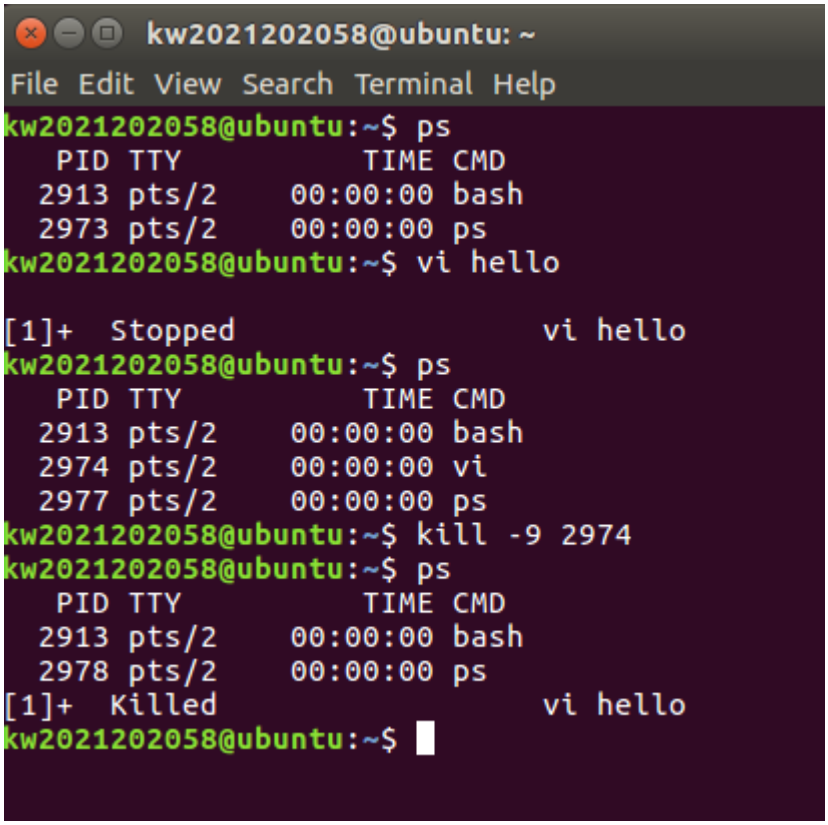
- exit



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

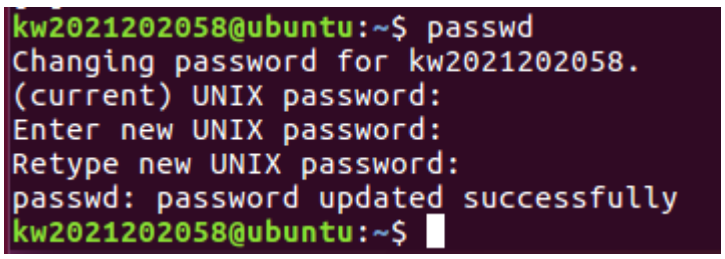
- kill


kw2021202058@ubuntu: ~
File Edit View Search Terminal Help
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my name
my nameTerminated
kw2021202058@ubuntu:~$
kw2021202058@ubuntu: ~
File Edit View Search Terminal Help
kw2021202058@ubuntu:~$ ps -e | tail
2401 ? 00:00:00 unity-scope-loa
2403 ? 00:00:00 unity-files-dae
2450 ? 00:00:00 kworker/0:0
2906 ? 00:00:22 gnome-terminal-
2913 pts/2 00:00:00 bash
2923 pts/2 00:00:10 yes
2938 pts/6 00:00:00 bash
2957 ? 00:00:02 kworker/u256:1
2962 pts/6 00:00:00 ps
2963 pts/6 00:00:00 tail
kw2021202058@ubuntu:~$ kill 6237
bash: kill: (6237) - No such process
kw2021202058@ubuntu:~$ kill 2923
kw2021202058@ubuntu:~$ ps -e | tail
2401 ? 00:00:00 unity-scope-loa
2403 ? 00:00:00 unity-files-dae
2450 ? 00:00:00 kworker/0:0
2906 ? 00:01:00 gnome-terminal-
2913 pts/2 00:00:00 bash
2938 pts/6 00:00:00 bash
2957 ? 00:00:05 kworker/u256:1
2964 ? 00:00:03 kworker/u256:2
2968 pts/6 00:00:00 ps



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



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$
```

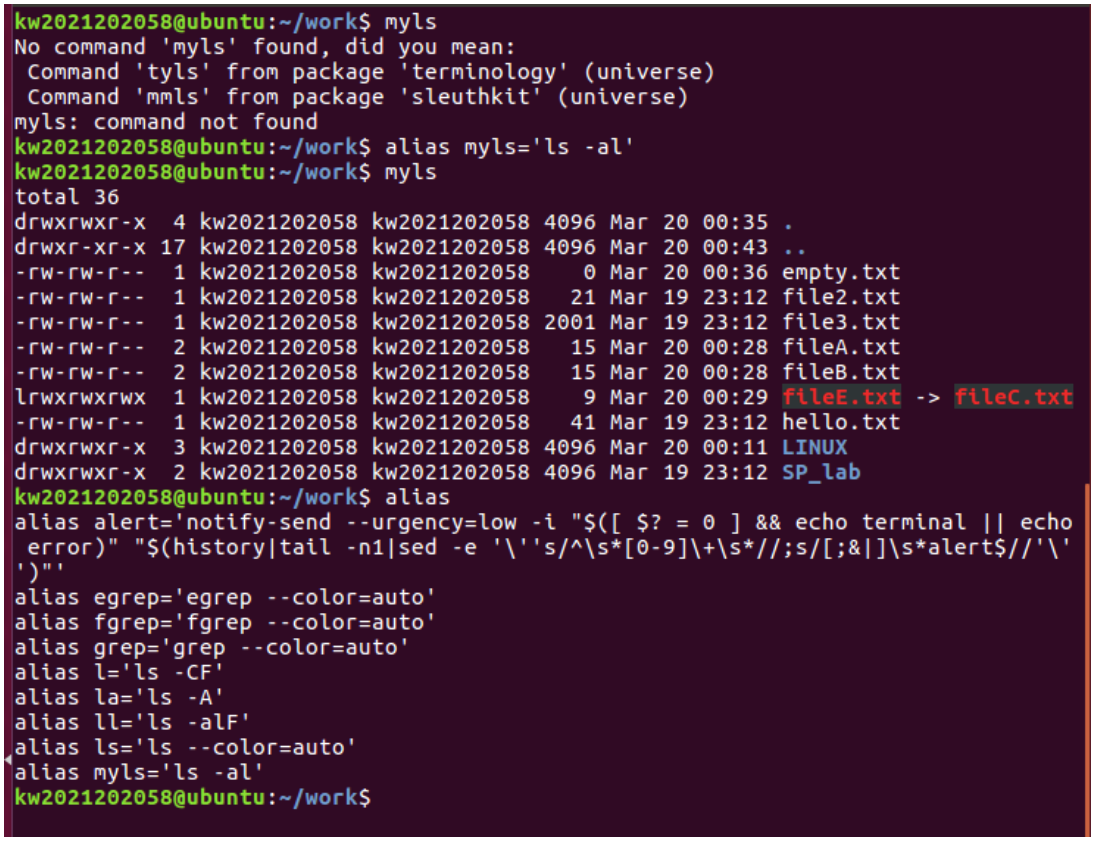
wc 명령어는 문서의 행, 단어, 문자의 개수를 계산하여 보여준다. 위 사진에서 볼 수 있듯이 hello.txt 파일의 행, 단어, 문자의 개수를 보여주는 것을 알 수 있다.

- echo

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

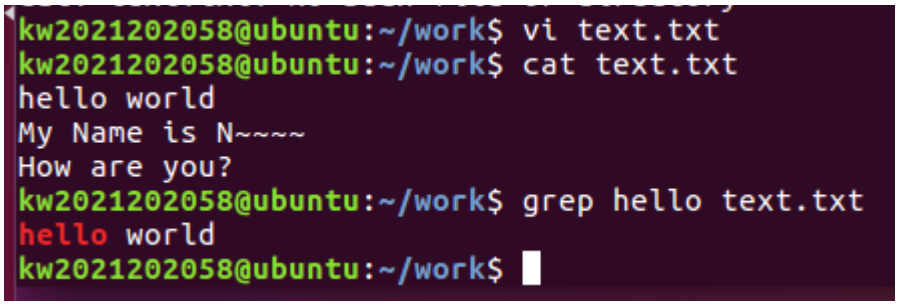
echo 명령어는 텍스트의 한 행을 보여준다. echo 명령어 뒤에 온 문자열을 출력한다.

- alias



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

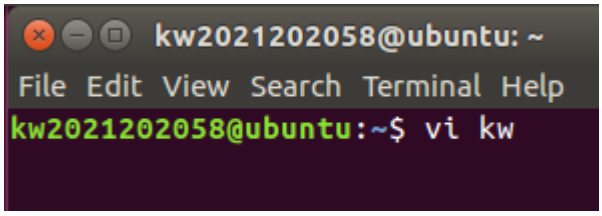
- grep

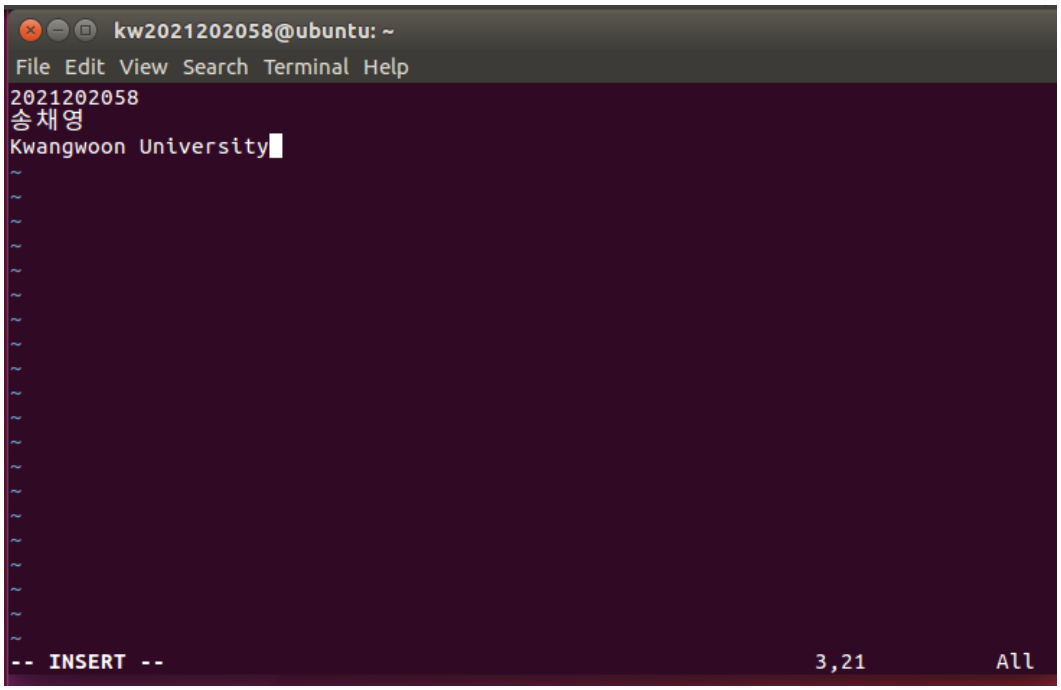


grep 명령어는 파일에서 찾고자 하는 어떤 단어를 찾아 표시한다. 위 사진에서 text.txt 파일이 없기 때문에 vi editor 를 통해 파일을 만들어준 후 cat 명령어를 통해 text.txt 파일을 출력해주었다. grep 명령어를 통해 hello 라는 단어를 찾도록 하였다.

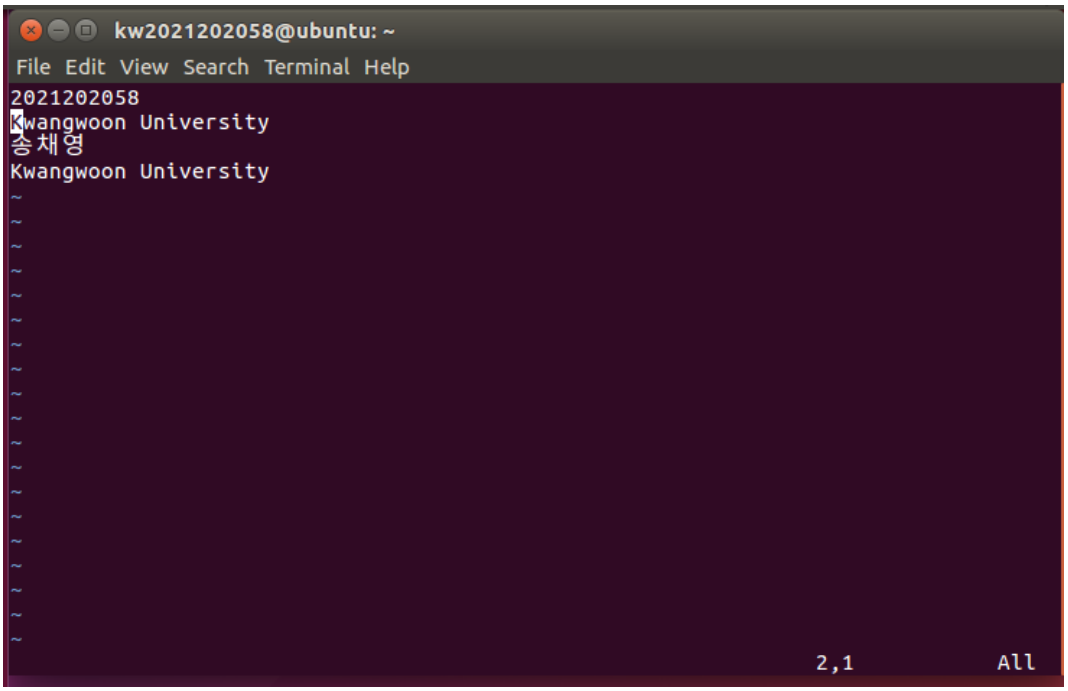
2-3. Linux_basedPrograming

- Vi editor

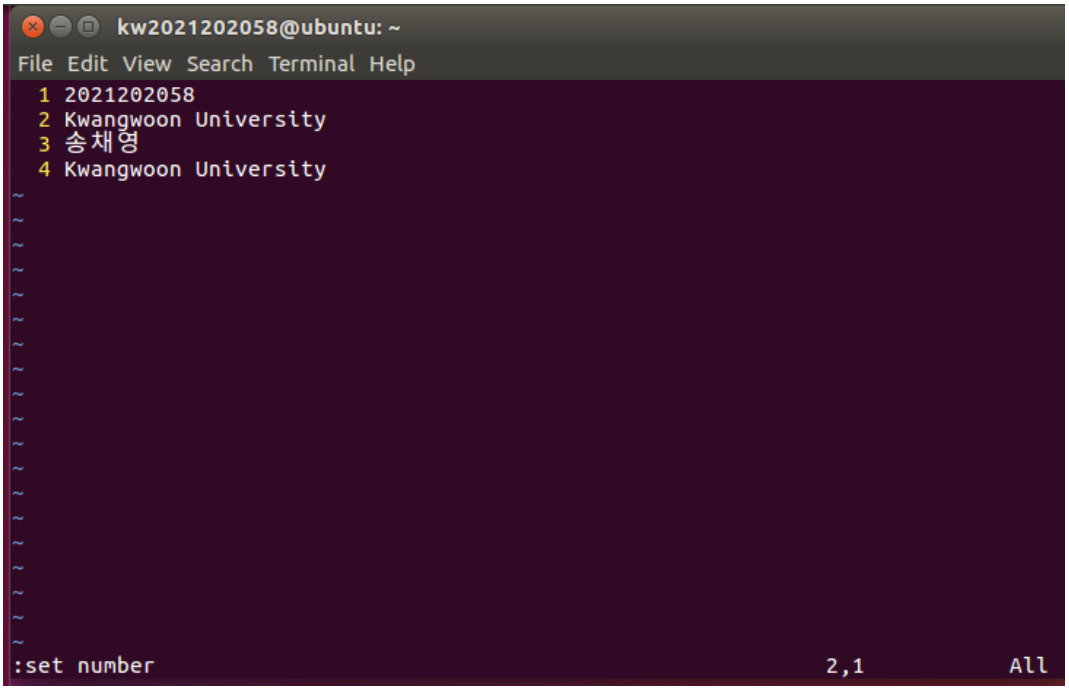




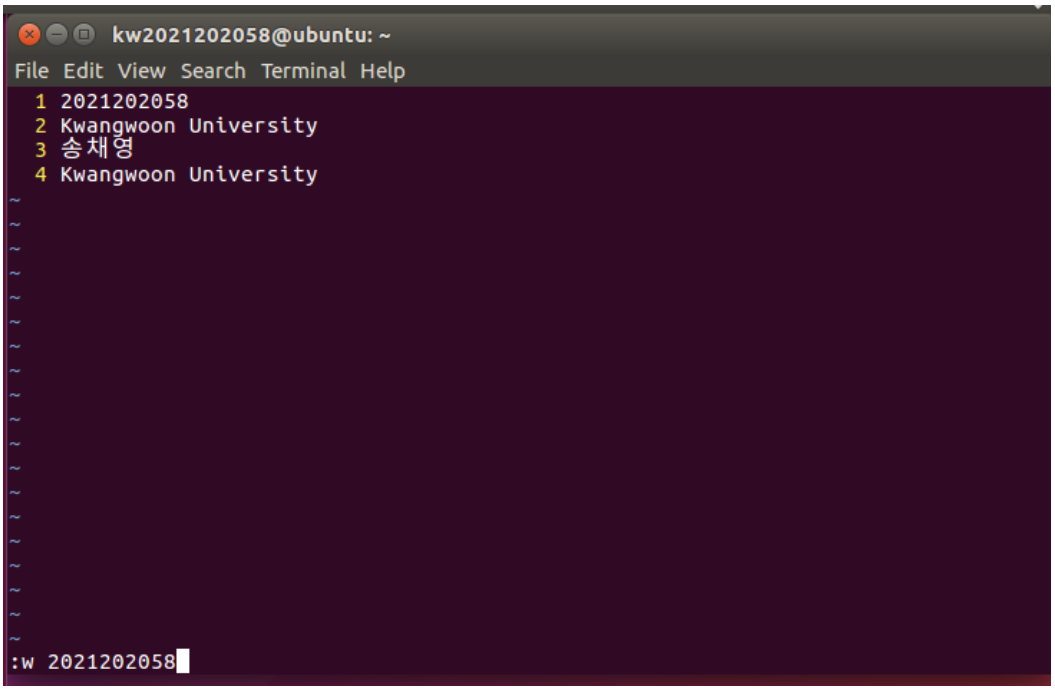
vi editor 을 통해 kw 라는 파일에 학번, 이름, kwangwoon University 를 입력하였다.



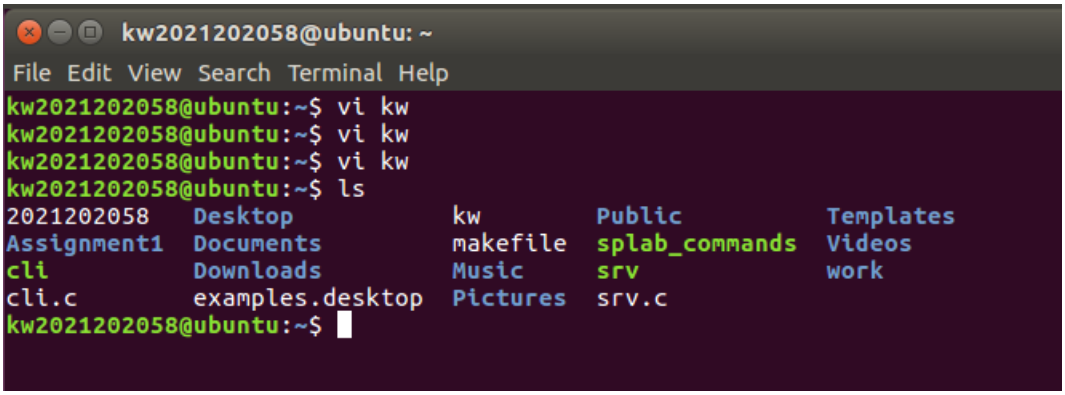
Kwangwoon University 행에 커서를 두고 yy 를 통해 그 행을 복사한 후 2021202058 행에 커서를 두고 p 를 눌러 복사한 문자열을 붙여넣었다.



명령모드에 set number 을 입력하여 편집기의 라인을 표시하였다.

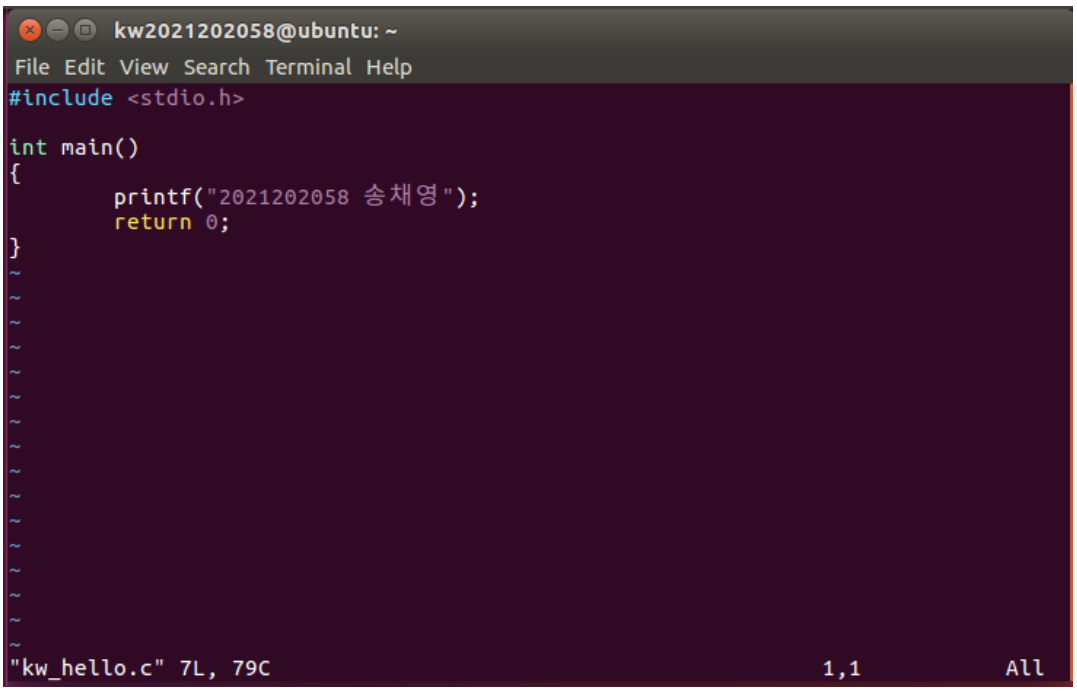


w 를 사용하여 학번으로 파일이름을 저장하고 wq 를 통해 저장하고 종료하였다.

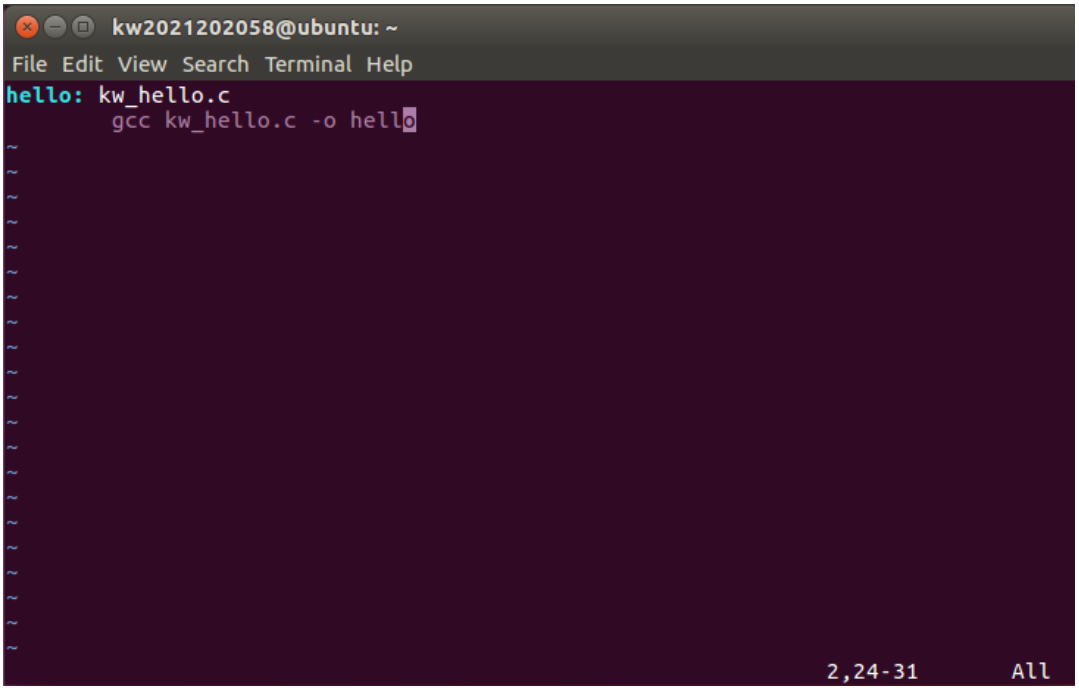


2021202058, 학번으로 파일이 저장된 것을 볼 수 있다.

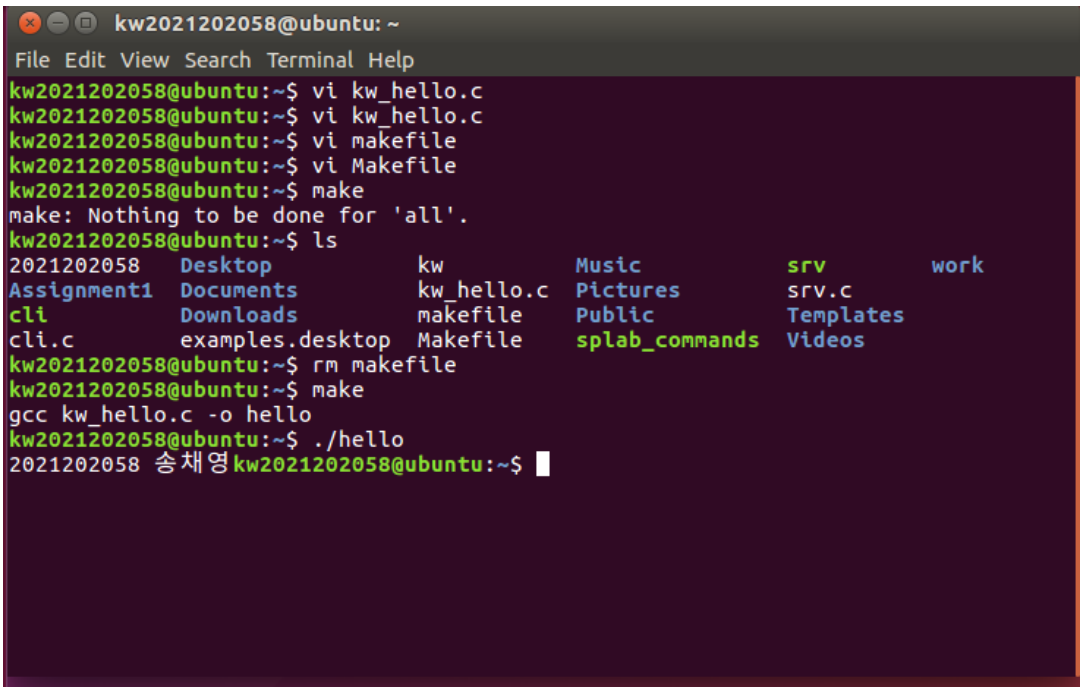
- Make



vi editor 를 통해 kw_hello.c 파일을 만든 후 학번과 이름을 출력하는 코드를 위의 사진과 같이 작성하였다.



vi editor 를 통해 makefile 을 만들었다.



make 명령어를 통해 컴파일을 진행하였는데, 지난 실습 시간에 실습한 makefile 로 인한 타겟 오류가 있어 파일을 지워주는 과정을 추가해주었다. 파일을 실행해보니 학번과 이름이 출력되는 것을 볼 수 있다.

3. 고찰

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 헷갈렸다. 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 헷갈렸다. 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 헷갈리고 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