시스템 프로그래밍

Basic 01~03

담당 교수님: 최상호 교수님
학번: 2021202045
이름: 김예은

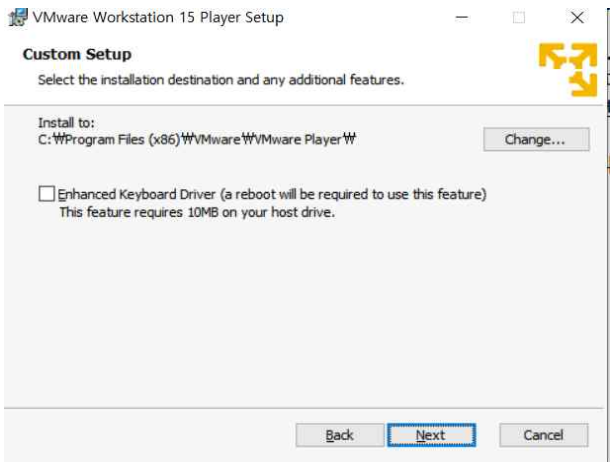
Ⅰ. Introduction

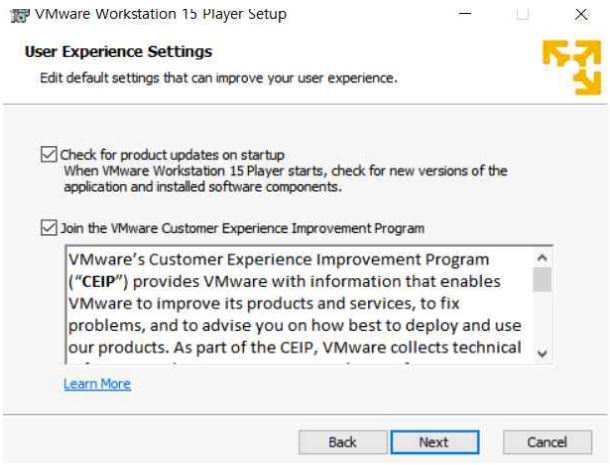
한 학기동안 진행될 시스템 프로그래밍 수업의 실습을 위해 vmware 15.5.7버전과 ubuntu 16.04.5버전을 다운 받아 실습 환경을 구축한다. 이러한 우분투 설치과정을 캡처하고 설명하는 과제가 Basic1 과제이며, 강의 자료에 나와 있는 Linux의 유용한 명령어들을 터미널 shell 창을 통해 실습해보며 명령어들의 기능을 배우고 캡처하는 것이 Basic2 과제이다. make를 활용하여 c파일을 컴파일하고, vi 명령어를 이용하여 편집기를 활용하는 과제가 Basic3이다.

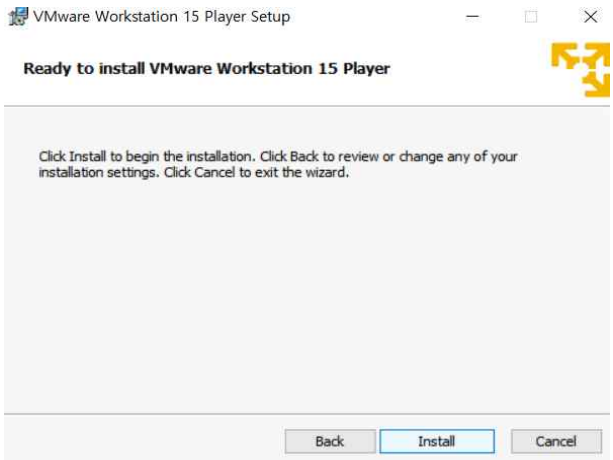
Ⅱ. 결과화면

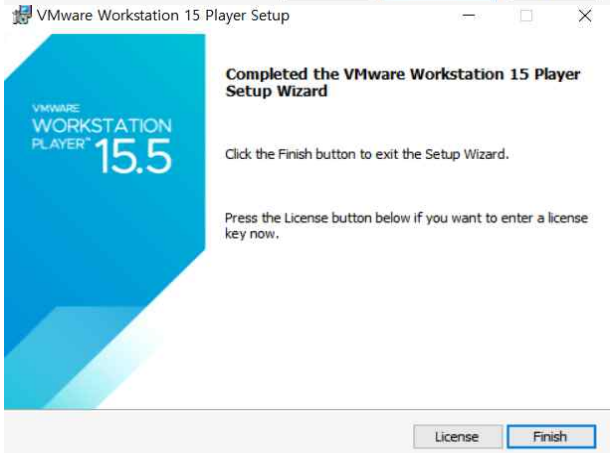
1. Basic-1(Ubuntu Installation)

- VMWare 설치과정





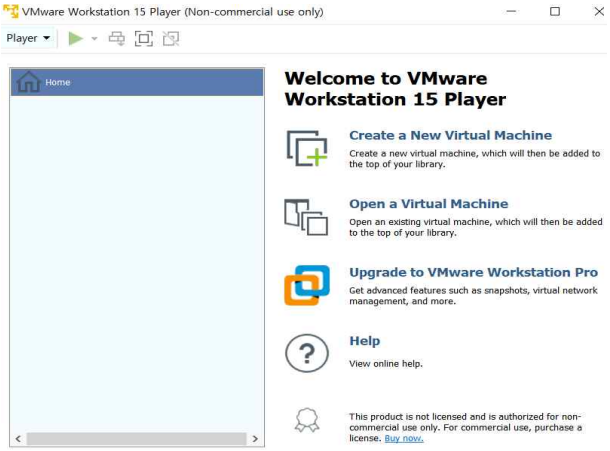




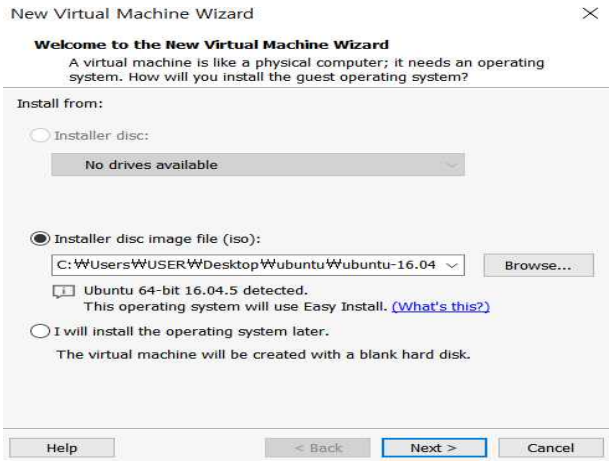
설치 파일을 실행한 후, 옵션 선택 과정에서 선택 변경 없이 단계를 진행하여 Install을 끝냈다.

- VMWare를 이용하여 Ubuntu 설치

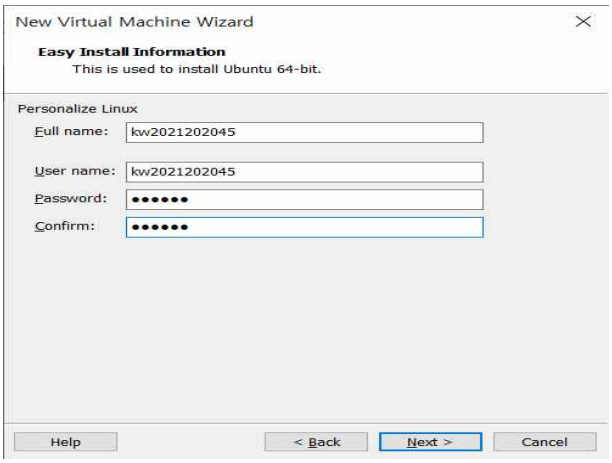
우선, VMWare를 실행하여 “Create a New virtual Machine”을 클릭한다.



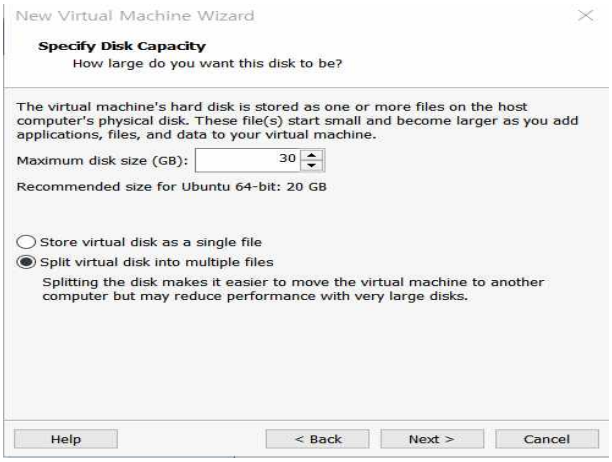
ubuntu iso파일 위치를 Browse를 통해 찾아 지정한다.



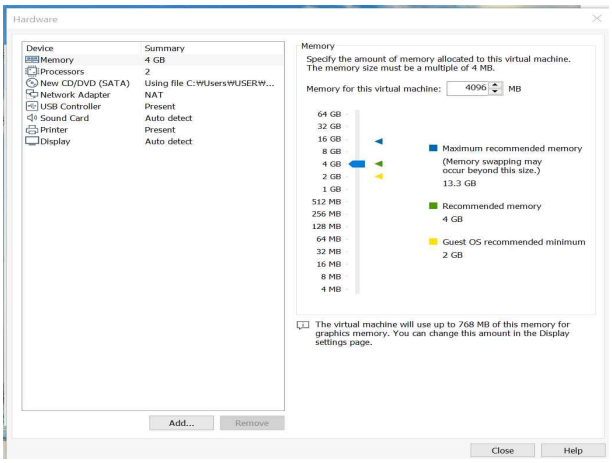
ubuntu에서 로그인 할 계정의 ID는 kw2021202045(kw+학번)으로 지정한다.



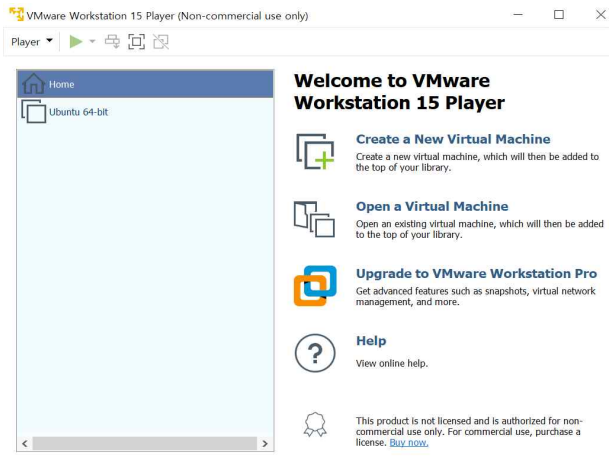
가상머신에 할당할 저장장치 크기를 설정한다. Disk size는 최소 30GB 이상 설정해야 하므로 추천 사이즈인 20보다 큰 30으로 지정해준다.

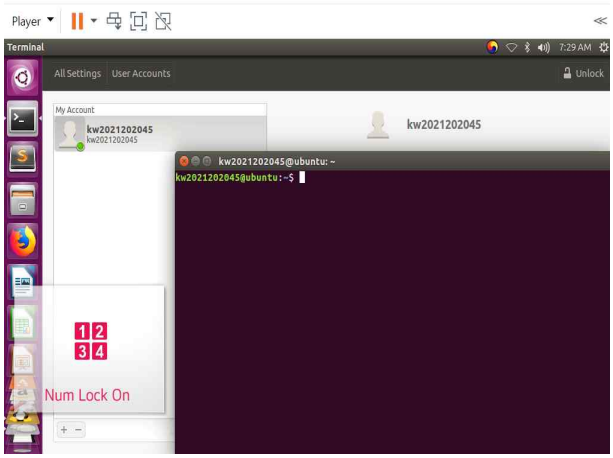


가상머신에 할당할 메인 메모리 크기, CPU크기, CPU core수를 설정한다. 본인 시스템 자체에서의 지원하는 크기는 4개의 core수를 가지고 있으므로 2개를 지정해준 후, 다운로드를 진행한다.



설치가 끝나면 다음과 같이 가상머신이 설치되었음을 볼 수 있고, 자주 사용할 terminal을 dock에 고정시켜준다.

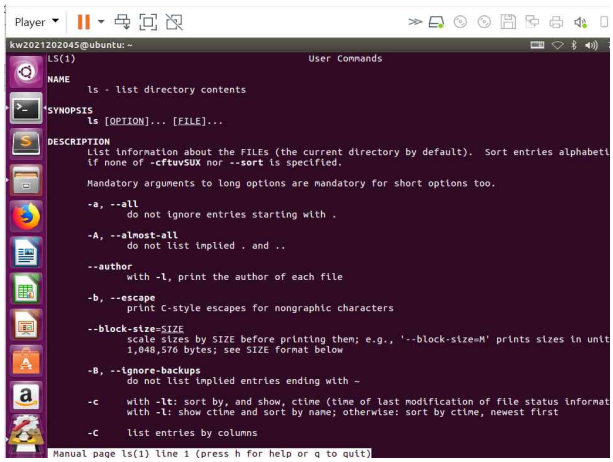




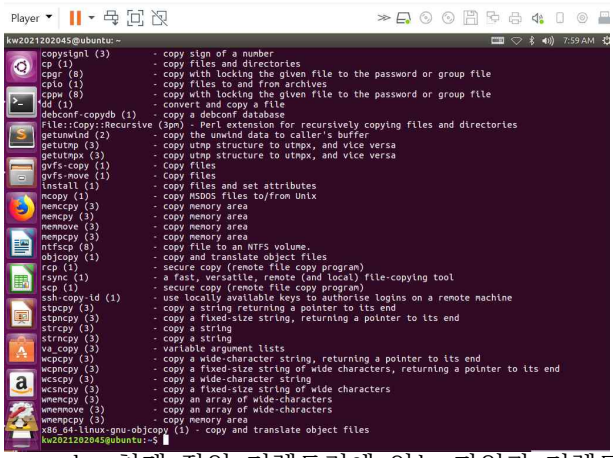
2.Basic-2(Linux Commands)

-man [option] name: name에 해당되는 명령어, 프로그램 사용법을 확인할 수 있다. q 키를 이용하여 manual에서 빠져나갈 수 있으며 h키를 이용하여 man 사용법을 확인할 수 있다.

(1-1) man ls: ls 명령어에 대한 설명을 하고 있다.

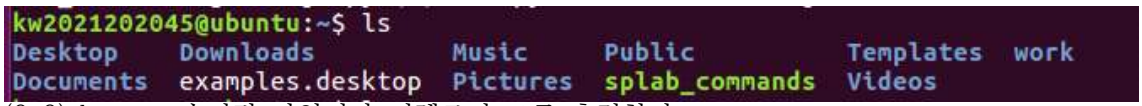


(1-2) man -k copy : -k옵션은 키워드 검색 기능을 제공하는데, 이 명령어를 사용하면 시스템에 설치된 모든 매뉴얼 페이지에서 지정한 키워드를 검색할 수 있다. 이 옵션을 이용하여 리눅스에서 특정 주제나 기능과 관련된 문서를 빠르게 찾을 수 있도록 도와준다. 즉, man -k copy의 경우 copy와 관련된 기능이나 문서를 찾아 보여준다. 다음의 캡처화면을 자세히 보면, 예를 들어 copy files를 하고 싶다면 cp, gvfs-copy, getutmpx 등이 있다고 가르쳐 주고 있다.

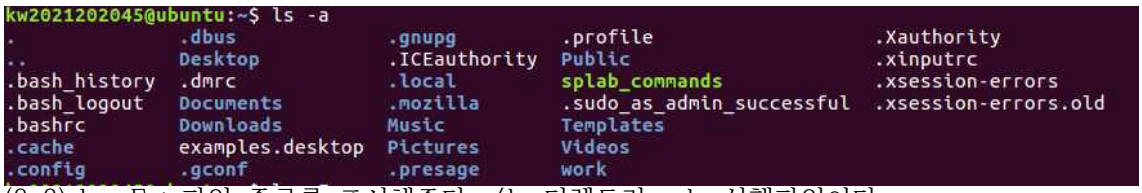


- ls: 현재 작업 디렉토리에 있는 파일과 디렉토리를 나열하는 기능을 제공한다.

(2-1) ls

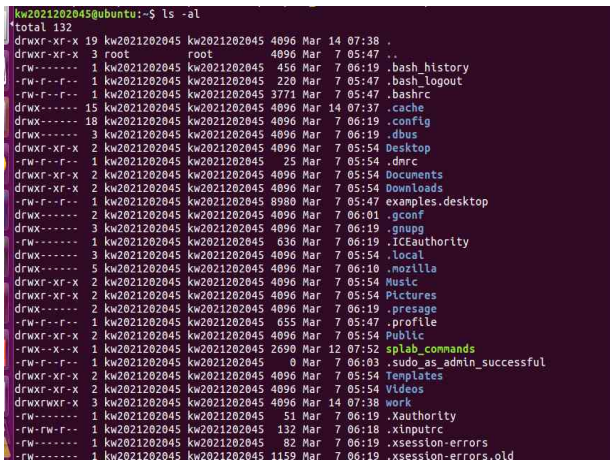


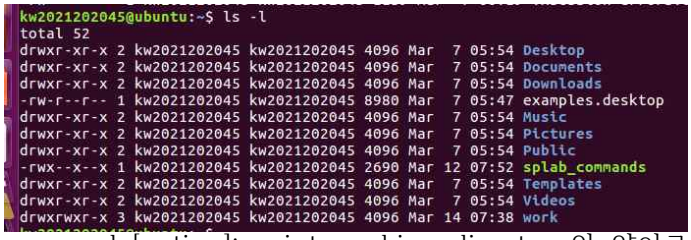
(2-2) ls -a : 숨겨진 파일이나 디렉토리 모두 출력한다.



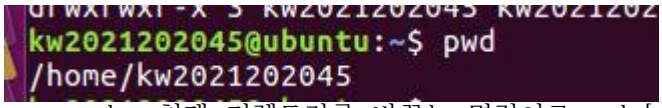
(2-3) ls -F : 파일 종류를 표시해준다. /는 디렉토리, *는 실행파일이다.

(2-4) ls -l : 현재 작업 디렉토리에 있는 파일의 종류, 권한, 소유자, 그룹, 파일 크기, 수정 시간 등의 정보를 확인할 수 있다. 다음은 a와l의 옵션을 같이 사용했기 때문에 숨겨진 파일에 대한 상세정보들도 같이 출력된 것을 볼 수 있다.





-pwd [option]: print working directory의 약어로, 현재 작업 디렉토리의 전체 경로를 출력한다. 다음은 현재 디렉토리의 전체 경로이다.



-cd : 현재 디렉토리를 바꾸는 명령어로, cd [dir]을 하면 해당 디렉토리로 이동하고, “cd .”의 경우 현재 디렉토리에 그대로, “cd ..”의 경우 parent directory로 이동한다. cd ~는 홈 디렉토리로 이동한다, cd -는 pwd와 같은 기능을 한다.

-cat: 파일의 내용을 출력하거나 파일을 합치는 기능을 제공한다. 합치는 기능을 할 때는 ‘>’ 기호를 사용한다. 다음은 file1.txt와 file2.txt의 파일 내용을 출력한다.

-chmod: 파일 접근 권한을 바꾸는 명령어이다. u는 user, g는 group, o는 other, a는 all을 뜻한다. +는 추가, -는 제거, =은 할당을 뜻한다. r은 read, w는 write, x는 execution을 뜻한다. OCTAL-MODE란 8진수 숫자 세 개로 user, group, other(순서중요)의 권한을 표현한 것으로, execute:1, write:2, read:4 각 숫자의 합으로 표현한다. 예를 들어, chmod 777 test란, 모든 대상에게 모든 권한을 부여하는 것이다. 다음 캡처화면에서 chmod 664 hello.txt란, user에는 r,w권한, group에도 r,w권한, other에게는 read 권한을 부여한다는 뜻이다. 또한 chmod u-w,g-w,o-r hello.txt 명령어 이후 hello.txt의 권한 부여가 “-rw-rw-r--”에서 “-r—r-----”로 바뀐 것을 볼 수 있다. 이것은 ls -l을 통해 파일의 상세 정보 보기 기능을 통해 확인할 수 있다.

```
kw2021202045@ubuntu:~/work$ ls -al
total 28
drwxrwxr-x  3 kw2021202045 kw2021202045 4096 Mar 14 07:38 .
drwxr-xr-x 19 kw2021202045 kw2021202045 4096 Mar 14 07:38 ..
-rw-rw-r--  1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r--  1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r--  1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-rw-r--  1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x  2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
kw2021202045@ubuntu:~/work$ chmod u-w,g-w,o-r hello.txt
kw2021202045@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-r--r----- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
kw2021202045@ubuntu:~/work$ chmod 644 hello.txt
kw2021202045@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
```

-mkdir : directory를 생성한다. mkdir SP_lecture -> SP_lecture라는 directory를 생성한다. 실행화면은 아래의 rmdir 명령어와 같이 사용했으므로 밑에 같이 첨부하겠다.

-rmdir : 빈 디렉토리를 삭제한다. 다음 캡처화면을 보면 ls -l을 통해 원래 없던 디렉토리가 mkdir을 통해 생성된 것을 볼 수 있고, rmdir을 통해 빈 디렉토리인 SP_lecture가 삭제되었다.

```
kw2021202045@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
kw2021202045@ubuntu:~/work$ mkdir SP_lecture
kw2021202045@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 08:30 SP_lecture
kw2021202045@ubuntu:~/work$ rmdir SP_Lecture
rmdir: failed to remove 'SP_Lecture': No such file or directory
kw2021202045@ubuntu:~/work$ rmdir SP_lecture
kw2021202045@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
```

-rm : 파일이나 디렉토리를 삭제하는 기능을 제공하는 명령어이다. 만약 rm fileA.txt를 한다면, fileA.txt가 삭제된다.

(9-1) rm -r [dir]: 디렉토리와 하위파일/디렉토리를 모두 삭제하는 명령어로, 재귀적으로 삭제한다. rm -r LINUX를 한 후 해당 이름을 가진 디렉토리가 삭제된 것을 확인할 수있다.

```
kw2021202045@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 15 18:34 LINUX
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
kw2021202045@ubuntu:~/work$ rm -r LINUX
kw2021202045@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
```

-cp: 파일과 디렉토리를 복사하는 것으로 다음 캡처화면을 보면, hello.txt를 hello_copy.txt에 복사를 하였고, cat을 통해 내용도 잘 복사되었는지 확인해주었다. “cp SP_lab/* .”의 경우 SP_lab 디렉토리에 있는 모든 파일과 디렉토리가 현재 작업 디렉토리로 복사되는 것이다. “.”의 경우 현재 작업 디렉토리를 나타내는 점이다.

```
kw2021202045@ubuntu:~/work$ cp hello.txt hello_copy.txt
kw2021202045@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 15 18:43 hello_copy.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
kw2021202045@ubuntu:~/work$ cat hello_copy.txt
hello world
My Name is N~~~
How are you?
kw2021202045@ubuntu:~/work$ cat hello.txt
hello world
My Name is N~~~
How are you?
```

```
kw2021202045@ubuntu:~/work$ cp SP_lab/* .
kw2021202045@ubuntu:~/work$ ls -l
total 32
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file1.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   21 Mar 14 07:38 file2.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045 2001 Mar 14 07:38 file3.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   15 Mar 15 18:45 fileA.txt
-rw-rw-r-- 1 kw2021202045 kw2021202045   15 Mar 15 18:45 fileC.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 15 18:43 hello_copy.txt
-rw-r--r-- 1 kw2021202045 kw2021202045   41 Mar 14 07:38 hello.txt
drwxrwxr-x 2 kw2021202045 kw2021202045 4096 Mar 14 07:38 SP_lab
kw2021202045@ubuntu:~/work$
```

-mv : 파일이나 디렉토리를 이동하거나 이름을 변경하는 기능을 제공한다. work 디렉토리 내에 있던 hello_copy.txt를 /home/kw2021202045/work/ex 위치로 이동시킨 것을 ex에 들어가 확인 할 수 있다. 그 다음 ex라는 디렉토리의 이름을 LINUX로 바꾼 것을 확인할 수 있다.

```
kw2021202045@ubuntu:~/work$ mv hello_copy.txt /home/kw2021202045/work/ex
kw2021202045@ubuntu:~/work$ ls
ex  file1.txt  file2.txt  file3.txt  fileA.txt  fileC.txt  hello.txt  SP_lab
kw2021202045@ubuntu:~/work$ cd ex
kw2021202045@ubuntu:~/work/ex$ ls
hello_copy.txt
kw2021202045@ubuntu:~/work/ex$ cd ..
kw2021202045@ubuntu:~/work$ mv ex LINUX
kw2021202045@ubuntu:~/work$ ls
file1.txt  file2.txt  file3.txt  fileA.txt  fileC.txt  hello.txt  LINUX  SP_lab
kw2021202045@ubuntu:~/work$
```

-ln: 파일이나 디렉토리에 대한 링크(link)를 생성하는 기능을 한다. inode란 리눅스 파일 시스템에서 파일이나 디렉토리를 식별하는 데 사용되는 구조체로, 파일이 생성되면 해당 파일은 새로운 inode를 할당받는다. 파일의 이름은 inode번호와 매핑되어 관리된다. ln은 하드링크와 심볼릭 링크 두가지 종류의 링크를 생성할 수 있는데, 하드링크는 원본 파일과 링크 파일이 동일한 inode를 가리키는 방식으로 링크를 생성한다. 심볼릭 링크는 원본 파일이나 디렉토리를 가리키는 심볼릭 링크 파일을 생성한다. 마치 바로가기 파일과 같은 것이다. 원본 파일의 크기와는 무관하다. 따라서 원본 파일이나 디렉토리가 삭제된다면 심볼릭 링크는 무의미해진다.

(12-1) ln(hard link)

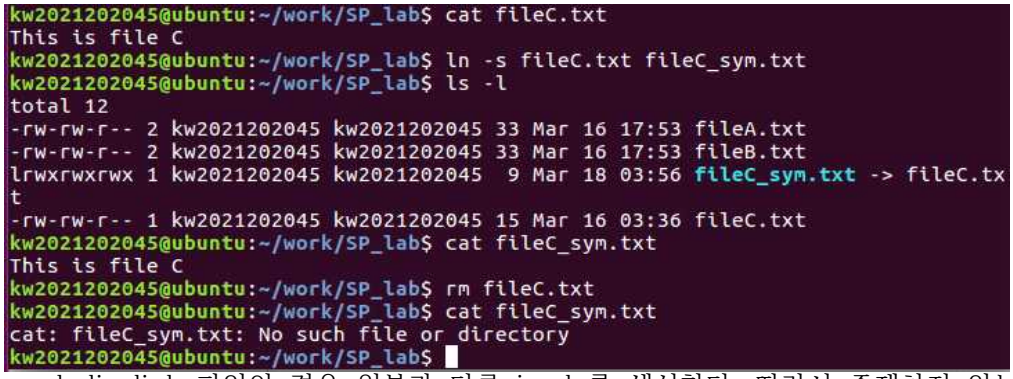
```
kw2021202045@ubuntu:~/work$ cd SP_lab
kw2021202045@ubuntu:~/work/SP_lab$ ls
fileA.txt  fileC.txt
kw2021202045@ubuntu:~/work/SP_lab$ cat fileA.txt
This is file A
kw2021202045@ubuntu:~/work/SP_lab$ ln fileA.txt fileB.txt
kw2021202045@ubuntu:~/work/SP_lab$ cat fileB.txt
This is file A
kw2021202045@ubuntu:~/work/SP_lab$ vi fileBtxt
kw2021202045@ubuntu:~/work/SP_lab$ vi fileB.txt
kw2021202045@ubuntu:~/work/SP_lab$ sudo apt-get install vim
```

위는 fileA.txt를 원본으로 가지는 하드링크 fileB.txt를 생성한 것이다. fileB.txt의 내용을 출력했을 때 fileA.txt와 같은 data를 가지는 것을 볼 수 있다. 그렇다면 fileB.txt를 편집기를 이용하여 내용을 고친다면

```
kw2021202045@ubuntu:~/work/SP_lab$ cat fileA.txt
This is file A
kw2021202045@ubuntu:~/work/SP_lab$ vi fileB.txt
kw2021202045@ubuntu:~/work/SP_lab$ cat fileA.txt
This is file B after the change.
kw2021202045@ubuntu:~/work/SP_lab$ cat fileB.txt
This is file B after the change.
kw2021202045@ubuntu:~/work/SP_lab$
```

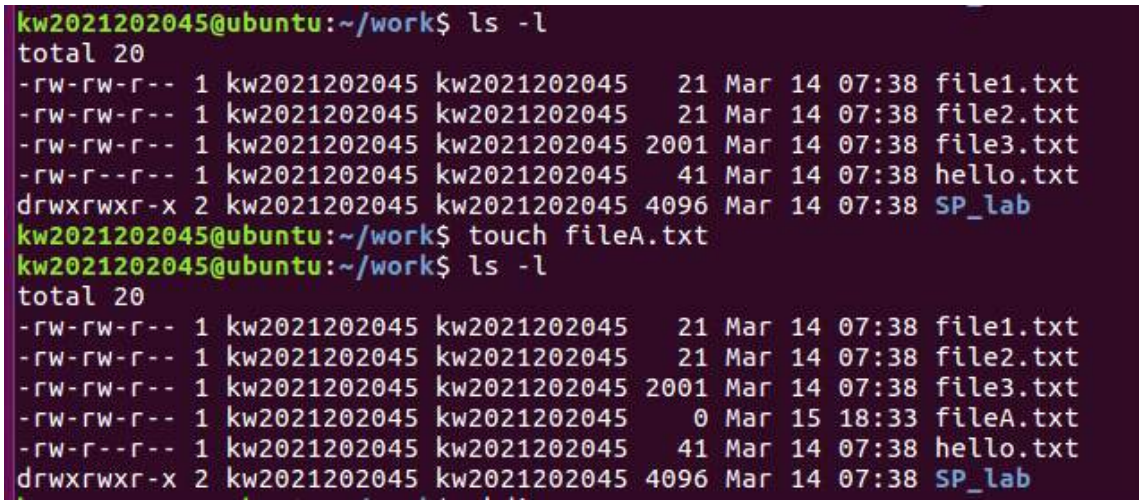
위와 같이 fileA.txt의 내용도 바뀐 것을 볼 수있다. 이는 원본 파일과 하드링크 파일이 같은 inode를 가리키기 때문이다. 따라서 존재하지 않는 파일에 대해 hard link를 작성할 수 없다.

(12-2) ln -s : 심볼릭 링크 파일 생성



symbolic link 파일의 경우 원본과 다른 inode를 생성한다. 따라서 존재하지 않는 파일이나 디렉토리에 대한 심볼릭 링크를 작성할 수 있으며 원본파일을 삭제해도 링크파일이 원본이 없다고 알려준다. 심볼릭 링크를 생성하는 명령어를 쓴 이후, 잘 생성 되었는 지 보기 위해 ls 명령어를 써준다. 심볼릭 링크를 걸게되면 접근권한부분 맨 왼쪽이 "l"로 표시된다.

-touch: 새로운 빈 파일을 만들거나 이미 존재하는 파일의 수정 시간을 변경하는 명령어이다. 원래는 fileA.txt가 work 디렉토리 내에 없었으나 touch fileA.txt 이후 ls -l을 통해 확인해보면 fileA.txt라는 빈 파일이 생겼음을 볼 수있다.



-ps: 리눅스에서 실행중인 process의 정보를 보여주는 명령어이다. 여러 가지 옵션을 통해 프로세스를 관리할 수 있다. ps명령어를 실행하면 현재 로그인한 사용자의 프로세스 목록을 보여주며, 각 프로세스에 대한 상세정보(프로세스 id, 실행시간, 메모리 사용량 등)을 보여주는데, 예를 들어 ps -ef는 시스템테서 실행 중인 모든 프로세스의 상세 정보를 보여준다.

```
kw2021202045@ubuntu:~/work$ ps
   PID TTY          TIME CMD
  2518 pts/1    00:00:00 bash
 24630 pts/1    00:00:00 ps
kw2021202045@ubuntu:~/work$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 17:07 ?        00:00:05 /sbin/init auto noprompt
root           2       0  0 17:07 ?        00:00:00 [kthreadd]
root           4       2  0 17:07 ?        00:00:00 [kworker/0:0H]
root           6       2  0 17:07 ?        00:00:00 [mm_percpu_wq]
root           7       2  0 17:07 ?        00:00:00 [ksoftirqd/0]
root           8       2  0 17:07 ?        00:00:01 [rcu_sched]
root           9       2  0 17:07 ?        00:00:00 [rcu_bh]
root          10       2  0 17:07 ?        00:00:00 [migration/0]
root          11       2  0 17:07 ?        00:00:00 [watchdog/0]
root          12       2  0 17:07 ?        00:00:00 [cpuhp/0]
root          13       2  0 17:07 ?        00:00:00 [cpuhp/1]
root          14       2  0 17:07 ?        00:00:00 [watchdog/1]
root          15       2  0 17:07 ?        00:00:00 [migration/1]
root          16       2  0 17:07 ?        00:00:00 [ksoftirqd/1]
root          18       2  0 17:07 ?        00:00:00 [kworker/1:0H]
root          19       2  0 17:07 ?        00:00:00 [kdevtmpfs]
root          20       2  0 17:07 ?        00:00:00 [netns]
root          21       2  0 17:07 ?        00:00:00 [rcu_tasks_kthre]
root          22       2  0 17:07 ?        00:00:00 [kauditd]
```

위의 출력 결과중 TTY란, 프로세스가 실행중인 터미널의 이름 또는 번호이다. 이 부분이 ?로 표시된다면, 이 프로세스는 터미널과 연결되어 있지 않은 백그라운드 프로세스를 나타내는 것이다. CMD란, 프로세스를 시작한 명령어 또는 프로그램이름이다. bash로 표시되는 것은, 해당 프로세스가 bash 셸에서 실행되고 있음을 나타낸다. bash는 대부분의 리눅스 시스템에서 사용되는 기본 셸이다.

-exit: 사용자가 현재 실행 중인 프로세스를 종료하는 명령어인데, 다음과 같이 exit을 입력하면 현재 셸 세션을 종료하게 된다.

```
kw2021202045@ubuntu:~$ ps
   PID TTY          TIME CMD
  2064 pts/6    00:00:00 bash
  2639 pts/6    00:00:00 ps
kw2021202045@ubuntu:~$ csh
% ps
   PID TTY          TIME CMD
  2064 pts/6    00:00:00 bash
  2640 pts/6    00:00:00 csh
  2643 pts/6    00:00:00 ps
% exit
% exit
kw2021202045@ubuntu:~$ ps
   PID TTY          TIME CMD
  2064 pts/6    00:00:00 bash
  2644 pts/6    00:00:00 ps
kw2021202045@ubuntu:~$
```

-passwd: 사용자의 암호를 변경하는 명령어이다. 아래의 캡처화면을 보면 먼저 현재 password를 확인하고, 새로운 password를 설정하게 한다. 예전의 password와 너무 비슷하

면 변경이 되지않고, 너무 단순해도 변경이 되지 않는 것을 볼 수 있다.

```
kw2021202045@ubuntu:~/work$ passwd
Changing password for kw2021202045.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
Bad: new and old password are too similar
kw2021202045@ubuntu:~/work$ passwd
Changing password for kw2021202045.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
Bad: new password is too simple
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

-uname: 여러 옵션과 함께 사용되며 시스템 정보를 출력해준다. uname만 사용할 시 시스템의 커널 버전 정보를 출력한다. 본인은 Linux를 사용하므로 다음과 같은 결과가 출력된다. “uname -r”의 경우, 시스템의 커널 버전을 출력하며, “uname -m”의 경우, 시스템의 하드웨어 플랫폼을 출력한다. “uname -a”는 시스템의 모든 정보를 출력한다.

```
kw2021202045@ubuntu:~/work$ uname
Linux
kw2021202045@ubuntu:~/work$ uname -r
4.15.0-29-generic
kw2021202045@ubuntu:~/work$ umane -m
No command 'umane' found, did you mean:
 Command 'umake' from package 'ubuntu-make' (universe)
umane: command not found
kw2021202045@ubuntu:~/work$ uname -m
x86_64
kw2021202045@ubuntu:~/work$ 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
```

즉, 본인이 사용하는 커널 버전은 위와 같고, 시스템의 하드웨어는 64비트 아키텍처를 지원하는 인텔과 AMD 프로세서를 사용한다는 뜻이다.

-wc: 같이 쓰인 파일명에 대한 파일의 행, 단어, 문자 수를 순서대로 보여준다. 다음을 보면 hello.txt에 대한 행의 수는 3, 단어는 9개, 41개의 문자가 쓰였음을 볼 수있다.

```
kw2021202045@ubuntu:~/work$ cat hello.txt
hello world
My Name is N~~~
How are you?
kw2021202045@ubuntu:~/work$ wc hello.txt
 3  9 41 hello.txt
kw2021202045@ubuntu:~/work$
```

-echo : echo 문자열의 경우 문자열을 출력한다. echo 명령어는 ‘$’기호와 함께 변수값을 출력하기도 한다. 다음과 같이 echo $HOME의 경우, 현재 사용자의 홈 디렉토리 경로를 보여준다. “echo -n”의 경우 문자열을 출력하되, 개행 문자는 출력되지 않는다.

```
kw2021202045@ubuntu:~/work$ echo helloworld
helloworld
kw2021202045@ubuntu:~/work$ echo $HOME
/home/kw2021202045
kw2021202045@ubuntu:~/work$ echo ~
/home/kw2021202045
kw2021202045@ubuntu:~/work$
```

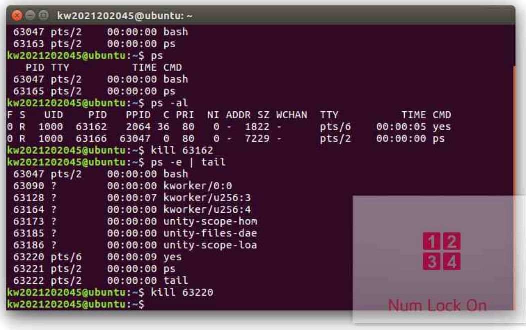
-alias: alias myls = 'ls -al'은 'myls'라는 명령어가 'ls -al'역할을 한다는 것이다. 즉 기존 명령어의 별명을 붙여주는 것인데 이렇게 정의된 별명은 새로운 쉘을 열 때마다 재정의 해줘야 한다. 그냥 alias만 쓴다면 별명 목록을 확인 할 수있다.

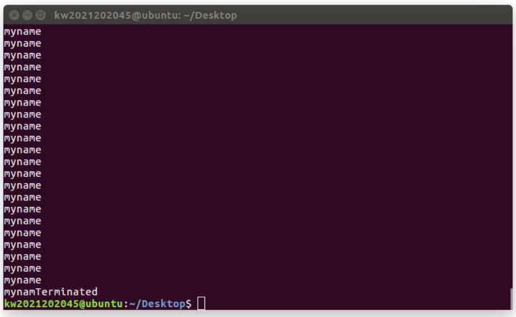
```
kw2021202045@ubuntu:~/work$ alias myls='ls -al'
kw2021202045@ubuntu:~/work$ myls
total 28
drwxrwxr-x  3 kw2021202045 kw2021202045 4096 Mar 16 03:36 .
drwxr-xr-x 16 kw2021202045 kw2021202045 4096 Mar 16 03:36 ..
-rw-rw-r--  1 kw2021202045 kw2021202045   21 Mar 16 03:36 file1.txt
-rw-rw-r--  1 kw2021202045 kw2021202045   21 Mar 16 03:36 file2.txt
-rw-rw-r--  1 kw2021202045 kw2021202045 2001 Mar 16 03:36 file3.txt
-rw-rw-r--  1 kw2021202045 kw2021202045   41 Mar 16 03:36 hello.txt
drwxrwxr-x  2 kw2021202045 kw2021202045 4096 Mar 16 03:36 SP_lab
kw2021202045@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 myls='ls -al'
```

-grep: 파일에서 특정 패턴을 검색하는 데 사용된다. 명령어와 같이 쓰인 특정패턴이 있는 모든 줄을 출력한다. 다음을 보면 hello.txt에서 hello가 있는 문자열, Name이 있는 문자열을 출력해준 것을 볼 수 있다.

```
kw2021202045@ubuntu:~/work$ ls
file1.txt  file2.txt  file3.txt  hello.txt  SP_lab
kw2021202045@ubuntu:~/work$ cat hello.txt
hello world
My Name is N~~~
How are you?
kw2021202045@ubuntu:~/work$ grep hello hello.txt
hello world
kw2021202045@ubuntu:~/work$ grep Name hello.txt
My Name is N~~~
kw2021202045@ubuntu:~/work$
```

-kill: 실행 중인 프로세스를 종료하는 기능을 제공한다. 다음의 실습 캡처 화면을 보면 yes myname을 입력하여 myname이 무한으로 출력되고 이때 또 다른 terminal창을 열어 ps -e | tail로 실행중인 yes 프로세서의 id를 kill해준다. |은 이전 명령어의 output을 다음 명령어의 input으로 주는 것이고, tail은 파일의 끝부분부터 10개의 행을 출력해준다.



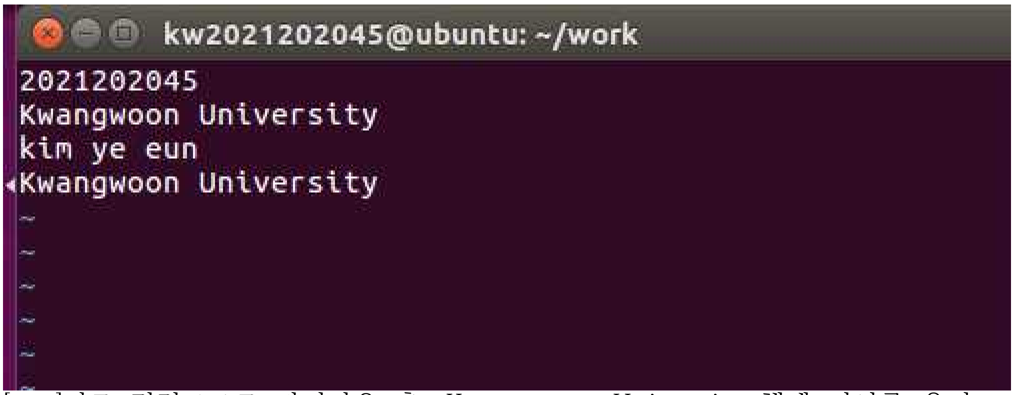


3.Basic-3(Linux_basedPrograming)

-Vi editor

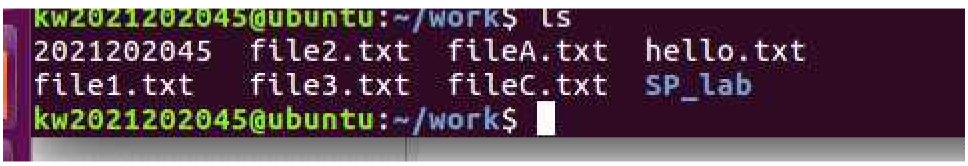
sudo apt-get install vim을 통해 vim 패키지 설치를 한 후, work directory 내에 vi 2021202045 명령어로 편집기를 연다. vi진입 시 기본모드에서 입력모드로 바꾸기 위해 [i]를 클릭 후, enter키로 줄 바꿈을 하여 본인 학번, 본인 이름, 제시된 문장을 쓴다.

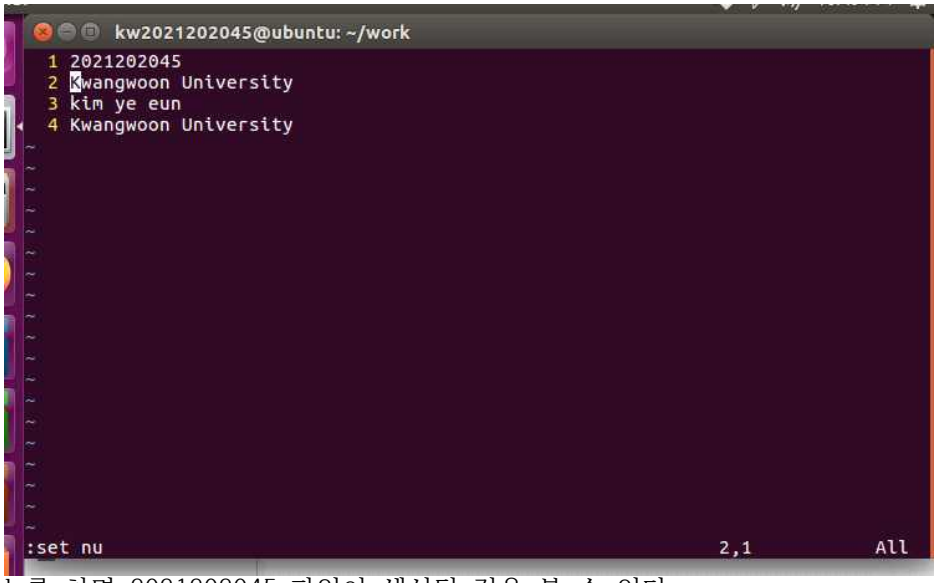
1라인 : 본인학번, 2라인: 본인 이름, 3라인: Kwangwoon University 입력



[esc]키로 명령모드로 빠져나온 후, Kwangwoon University 행에 커서를 옮겨 yy를 입력한다. yy는 커서가 있는 행을 복사하는 키이다. 그 다음 학번이 있는 행에 커서를 옮겨 p를 입력한다. p는 커서 아래 복사된 문자열을 붙인다.

명령모드에서 :set nu를 입력하여 편집기에 라인을 표시한다. 그 다음 :w 2021202045를 입력하여 2021202045(학번)을 파일명으로 저장한다.

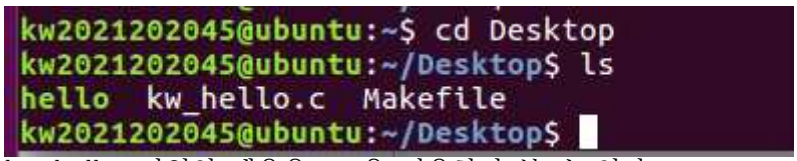




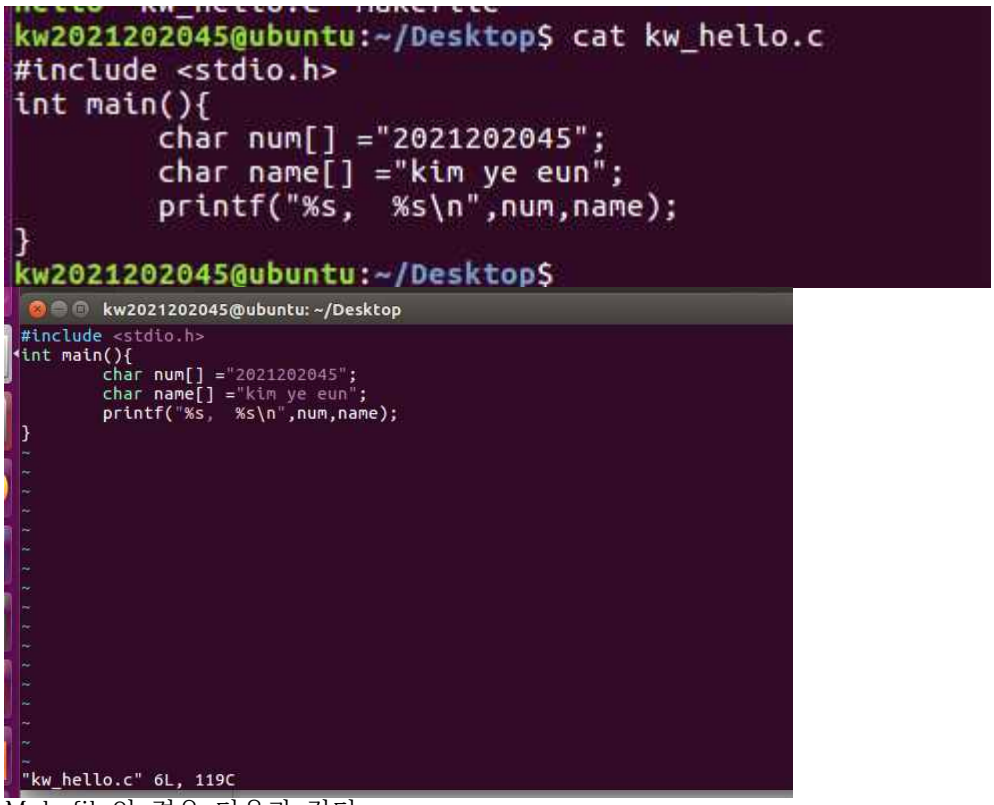
ls를 하면 2021202045 파일이 생성된 것을 볼 수 있다.

-Make

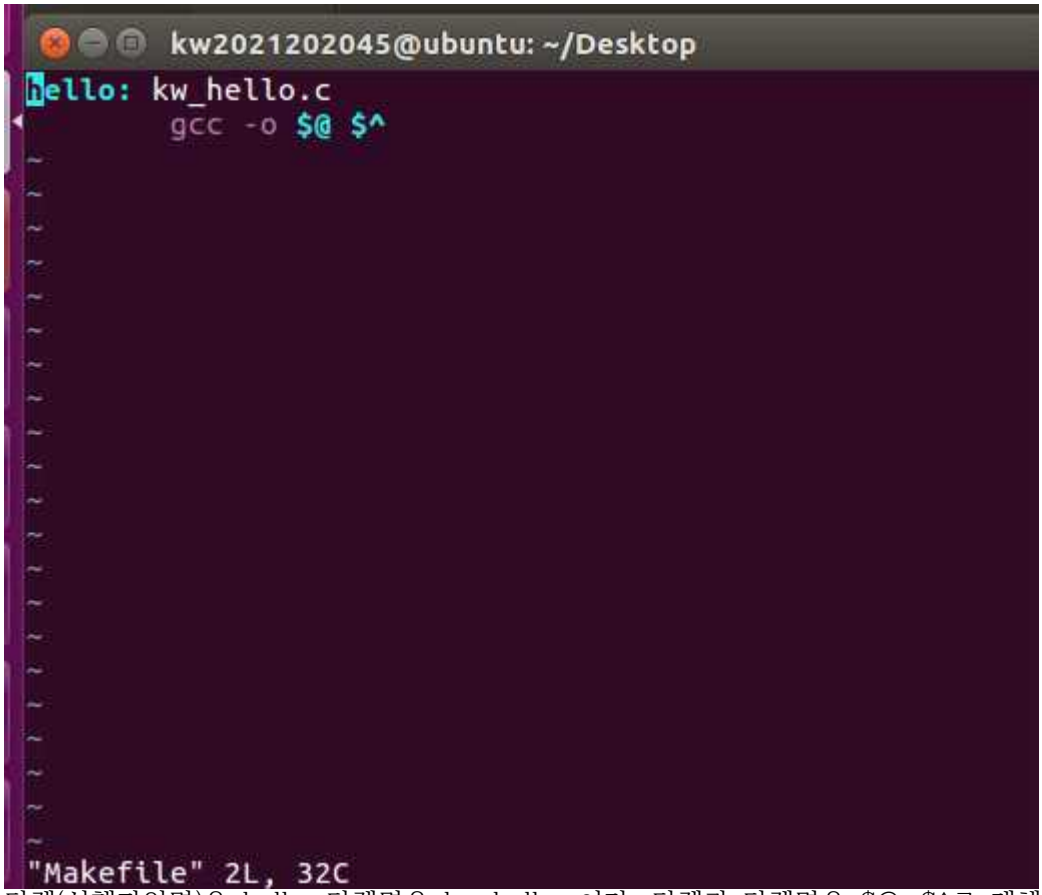
본인은 Desktop 디렉토리에 kw_hello.c 파일을 생성했다.



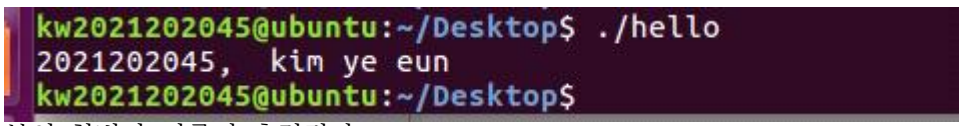
kw_hello.c파일의 내용을 cat을 이용하여 볼 수 있다.



Makefile의 경우 다음과 같다.



타겟(실행파일명)은 hello, 타겟명은 kw_hello.c이다. 타겟과 타겟명은 $@, $^로 대체한다.
실행결과는 다음과 같다.



본인 학번과 이름이 출력된다.

Ⅲ. 고찰

맨처음에 linux를 깔고 csh가 install 되지 않아서 오류 메시지를 해석하고 해결해보려고 했으나 이틀 내내 해결되지 않아서 다시 지웠다 깔아서 해결했다. 또한, 리눅스를 설치하고 재부팅을 해줘야 파일 복사 붙여넣기 기능이 된다는 것도 처음 알았다. 터미널을 하나 더 열 수 있다는 사실을 알고, kill명령어를 했다. 작년 데구설 수업시간에는 조교님이 주신 Makefile로 make를 진행했는데 이번에 Makefile작성법을 배우면서 직접 Makefile을 작성해보니 흥미로웠다. c언어는 많이 까먹었는데 앞으로의 과제를 위해 미리 복습해야겠다고 다짐했다.

Ⅳ. Reference

생략