

# 시스템 프로그래밍 실습 Report

실습 제목: Basic

실습일자: 2022년 3월 10일 (목)

제출일자: 2022년 3월 23일 (수)

학 과: 컴퓨터정보공학부

담당교수: 최상호 교수님

실습분반: 목요일 7,8교시

학 번: 2018202065

성 명: 박 철 준

- Introduction

1. 제목

Bazic

2. 목적

가. Ubuntu Installation – Vmware 설치과정 및 Ubuntu를 설치하는 과정을 수행하고 이를 설명할 수 있어야한다. 여기서 Vmware는 가상머신의 한 종류로 가상 머신(영어: virtual machine, VM)은 컴퓨팅 환경을 소프트웨어로 구현한 것이다. 즉 컴퓨터 시스템을 가상 현실화하는 소프트웨어이다. 가상 머신상에서 운영 체제나 응용 프로그램을 설치 및 실행할 수 있다. Vmware 말고도 다른 기능들이 있는 여러 종류의 가상 머신들이 있다.

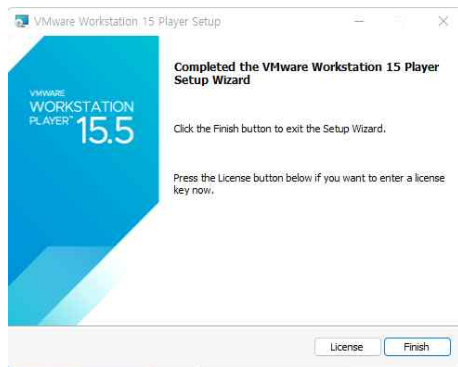
나. Linux Commands – Linux 명령어들을 사용하고, 이를 설명하여야한다.

다. Linux\_basedProgramming – vi, make를 사용하고, 이를 설명하여야한다. 여기서 Vi는 유닉스 환경에서 가장 많이 쓰이는 문서 편집기이다. 1976년 빌 조이가 초기 BSD 릴리즈에 포함될 편집기로 만들었다. vi라는 이름은 한 줄씩 편집하는 줄 단위 편집기가 아니라 한 화면을 편집하는 비주얼 에디터(visual editor)라는 뜻에서 유래했다. 간결하면서도, 강력한 기능으로 열광적인 사용자가 많다. 또한 make는 파일 간의 종속관계를 파악하여 Makefile에 적힌 대로 컴파일러에 명령하여 SHELL 명령이 순차적으로 실행될 수 있게 하는 프로그램 빌드 도구이다.

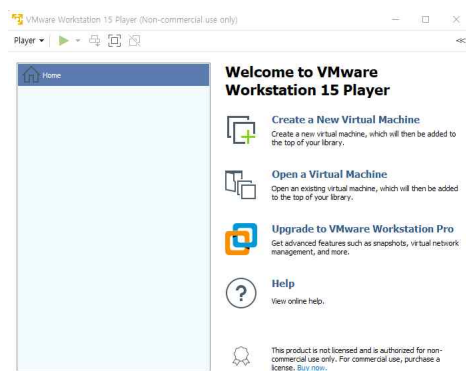
## ● 실습 결과

### 1. Vmware 및 Ubuntu 설치

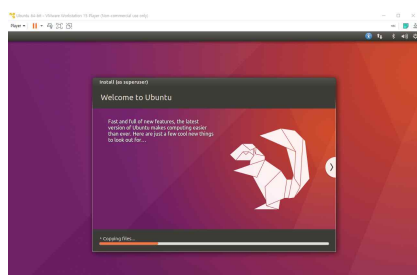
아래는 vmware 설치파일을 실행 후 마친 모습이다.



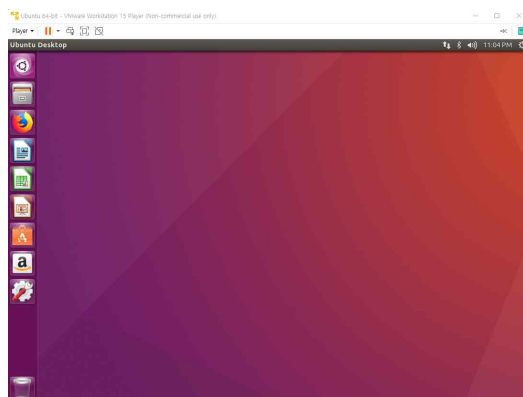
아래는 vmware를 실행한 모습이다.



아래는 우분투 iso파일을 이용하여 vmware를 통해 우분투를 설치하는 과정이다.



아래는 우분투가 설치가 완료된 모습입니다.



## 2. Linux Commands 사용 및 설명

-man 명령어

```
kw2018202065@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 specified.

  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 ~

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

man 명령어는 manual명령어로서 위의 사진을 참고하면 ls명령어에 대한 설명이 적힌 문서를 보여주는 명령어이다.(man ls)

아래는 mkdir이 가지고 있는 섹션을 보기위해 man -k [명령어] 옵션을 이용한 모습이다.

```
kw2018202065@ubuntu: ~
kw2018202065@ubuntu:~$ man -k mkdir
gvfs-mkdir (1) - Create directories
mkdir (1) - make directories
mkdir (2) - create a directory
mkdirat (2) - create a directory
kw2018202065@ubuntu:~$
```

man이 보여줄 수 있는 섹션은 총 8개로

1번째는 일반적인 명령어에 대한 설명을 보여준다.

2번째는 system call에 대한 매뉴얼을 보여준다.

3번째는 c언어의 라이브러리 함수에 대한 매뉴얼을 보여준다.

4번째는 장치나 특수파일, 드라이브에 대한 매뉴얼을 보여준다.

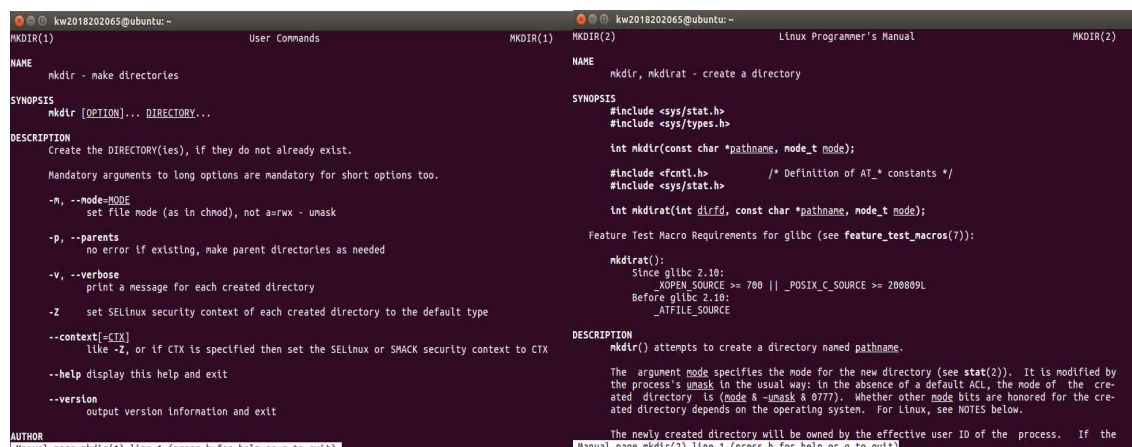
5번째는 환경설정 파일들에 대한 매뉴얼을 보여준다.

6번째는 게임과 같은 프로그램들에 대한 매뉴얼을 보여준다.

7번째는 리눅스의 파일 표준이나 규칙, 프로토콜, 시그널에 대한 매뉴얼들을 보여준다.

8번째는 관리자들이 사용하는 명령어나 데몬에 대한 매뉴얼들을 보여준다.

아래는 간단하게 man 1 mkdir 과 man 2 mkdir을 사용하여 명령어의 일반적인 설명과 system call에 대한 매뉴얼들을 확인해본 모습이다.



```
kw2018202065@ubuntu:~$ man 1 mkdir
NAME
  mkdir - make directories

SYNOPSIS
  mkdir [OPTION]... DIRECTORY...

DESCRIPTION
  Create the DIRECTORY(ies), if they do not already exist.

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

  -m, --mode=MODE
    set file mode (as in chmod), not a=rwx - umask

  -p, --parents
    no error if existing, make parent directories as needed

  -v, --verbose
    print a message for each created directory

  -Z
    set SELinux security context of each created directory to the default type

  --context[=CTX]
    like -Z, or if CTX is specified then set the SELinux or SMACK security context to CTX

  --help
    display this help and exit

  --version
    output version information and exit

AUTHOR
  Manual page mkdir(1) line 1 (press h for help or q to quit)

kw2018202065@ubuntu:~$ man 2 mkdir
NAME
  mkdir, mkdirat - create a directory

SYNOPSIS
  #include <sys/stat.h>
  #include <sys/types.h>

  int mkdir(const char *pathname, mode_t mode);

  #include <fcntl.h> /* Definition of AT_* constants */
  #include <sys/stat.h>

  int mkdirat(int dirfd, const char *pathname, mode_t mode);

  Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

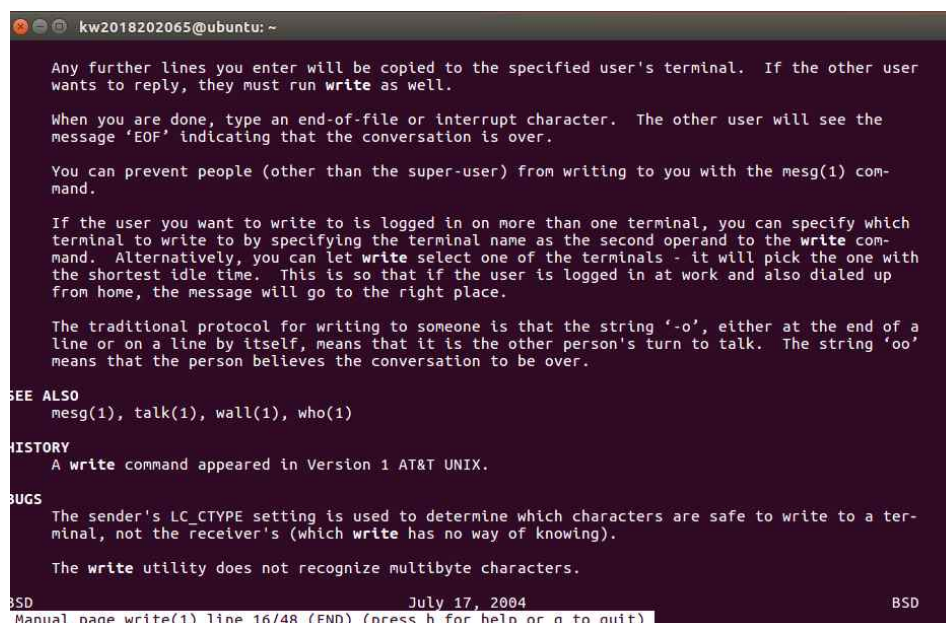
  mkdirat():
    Since glibc 2.10:
      _XOPEN_SOURCE >= 700 || _POSIX_C_SOURCE >= 200809L
    Before glibc 2.10:
      _ATFILE_SOURCE

DESCRIPTION
  mkdir() attempts to create a directory named pathname.

  The argument mode specifies the mode for the new directory (see stat(2)). It is modified by the process's umask in the usual way: in the absence of a default ACL, the mode of the created directory is (mode & ~umask & 0777). Whether other mode bits are honored for the created directory depends on the operating system. For Linux, see NOTES below.

  The newly created directory will be owned by the effective user ID of the process. If the
  Manual page mkdir(2) line 1 (press h for help or q to quit)
```

또한 man -a [명령어]를 통해 모든 매뉴얼을 볼 수 있다. 아래는 man -a write를 통해 write에 대한 모든 매뉴얼을 출력한 모습이다.



```
kw2018202065@ubuntu:~$ man -a write

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) com-
mand.

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 com-
mand. 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 ter-
  minal, not the receiver's (which write has no way of knowing).

  The write utility does not recognize multibyte characters.

BSD
  July 17, 2004
  Manual page write(1) line 16/48 (END) (press h for help or q to quit)
```

-ls 명령어

ls 명령어는 디렉토리에 있는 파일들을 볼 수 있는 명령어이다. 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~  
kw2018202065@ubuntu:~$ ls  
Desktop Downloads Music Public Templates work  
Documents examples.desktop Pictures splab_commands Videos  
kw2018202065@ubuntu:~$ ls -a  
.  
..  
.cache Documents .gnupg Music splab_commands .Xauthority  
.config Downloads .ICEauthority Pictures Templates .xsession-errors  
.bash_logout Desktop examples.desktop .local .profile Videos .xsession-errors.old  
.bashrc .dmrc .gconf .mozilla Public work  
kw2018202065@ubuntu:~$ ls -F  
Desktop/ Downloads/ Music/ Public/ Templates/ work/  
Documents/ examples.desktop Pictures/ splab_commands* Videos/  
kw2018202065@ubuntu:~$ ls -al  
total 116  
drwxr-xr-x 17 kw2018202065 kw2018202065 4096 Mar 15 00:13 .  
drwxr-xr-x 3 root root 4096 Mar 14 22:54 ..  
-rw-r--r-- 1 kw2018202065 kw2018202065 220 Mar 14 22:54 .bash_logout  
-rw-r--r-- 1 kw2018202065 kw2018202065 3771 Mar 14 22:54 .bashrc  
drwx----- 14 kw2018202065 kw2018202065 4096 Mar 15 00:07 .cache  
drwx----- 14 kw2018202065 kw2018202065 4096 Mar 14 23:04 .config  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:08 Desktop  
-rw-r--r-- 1 kw2018202065 kw2018202065 25 Mar 14 23:02 .dmrc  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Documents  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Downloads  
-rw-r--r-- 1 kw2018202065 kw2018202065 8980 Mar 14 22:54 examples.desktop  
drwx----- 2 kw2018202065 kw2018202065 4096 Mar 15 00:11 .gconf  
drwx----- 3 kw2018202065 kw2018202065 4096 Mar 15 00:07 .gnupg  
-rw----- 1 kw2018202065 kw2018202065 954 Mar 15 00:07 .ICEauthority  
drwx----- 3 kw2018202065 kw2018202065 4096 Mar 14 23:02 .local  
drwx----- 5 kw2018202065 kw2018202065 4096 Mar 14 23:03 .mozilla  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Music  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Pictures  
-rw-r--r-- 1 kw2018202065 kw2018202065 655 Mar 14 22:54 .profile  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Public  
-rw-x--x--x 1 kw2018202065 kw2018202065 2690 Mar 13 00:58 splab_commands  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Templates  
drwxr-xr-x 2 kw2018202065 kw2018202065 4096 Mar 14 23:02 Videos  
drwxrwxr-x 3 kw2018202065 kw2018202065 4096 Mar 15 00:13 work  
-rw----- 1 kw2018202065 kw2018202065 51 Mar 15 00:07 .Xauthority  
-rw----- 1 kw2018202065 kw2018202065 82 Mar 15 00:07 .xsession-errors  
-rw----- 1 kw2018202065 kw2018202065 1241 Mar 14 23:09 .xsession-errors.old  
kw2018202065@ubuntu:~$
```

위 사진에 쓰인 명령어를 설명하면 다음과 같다.

1. ls 는 숨겨진 파일을 제외한 모든 파일을 보여준다.
2. ls -a은 숨겨진 파일을 포함하여 모든 파일을 보여준다.
3. ls -F는 파일의 종류를 표시해준다.(디렉토리는 /, 실행파일은 \*을 사용한다.)
4. ls -al은 파일의 정보를 자세히 출력해준다.(권한, 사용자, 용량, 마지막 수정날짜 등)

- pwd 명령어

```
kw2018202065@ubuntu: ~  
kw2018202065@ubuntu:~$ pwd  
/home/kw2018202065  
kw2018202065@ubuntu:~$
```

pwd명령어는 현재 사용자가 위치해 있는 디렉토리를 출력해준다.

-cd 명령어

```
kw2018202065@ubuntu: ~/work  
kw2018202065@ubuntu:~$ pwd  
/home/kw2018202065  
kw2018202065@ubuntu:~$ ls  
Desktop Documents Downloads examples.desktop Music Pictures Public splab_commands Templates Videos work  
kw2018202065@ubuntu:~$ cd work  
kw2018202065@ubuntu:~/work$ pwd  
/home/kw2018202065/work  
kw2018202065@ubuntu:~/work$ cd .  
kw2018202065@ubuntu:~/work$ cd ..  
kw2018202065@ubuntu:~$ cd work  
kw2018202065@ubuntu:~/work$ cd ~  
kw2018202065@ubuntu:~$ cd -  
/home/kw2018202065/work  
kw2018202065@ubuntu:~/work$
```

위는 cd명령어의 사용 예시로 cd명령어는 현재 디렉토리의 위치를 바꿔줄 수 있는 명령어



이다.

먼저 cd work를 통해 work 폴더로 접속 후 cd .을 하게 되면 현재 디렉토리에 머물 수 있고 cd ..을 하게 되면 상위 디렉토리로 이동할 수 있게 해준다. cd -은 이전경로로의 이동이다.

#### -cat 명령어

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ cat file1.txt
Hello This is file 1
kw2018202065@ubuntu:~/work$ cat file2.txt
Hello This is file 2
kw2018202065@ubuntu:~/work$ cat file1.txt file2.txt
Hello This is file 1
Hello This is file 2
kw2018202065@ubuntu:~/work$
```

위의 cat명령어의 사용 예시로 cat명령어는 파일을 연결하고 출력하는 명령어이다.

먼저 cat file1.txt를 통하여 텍스트파일안에 있는 글자를 출력하고

cat file1.txt file2.txt를 통하여 텍스트파일 두 개를 연결하여 출력한 모습이다.

#### - chmod 명령어

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls -al
total 28
drwxrwxr-x 3 kw2018202065 kw2018202065 4096 Mar 15 00:13 .
drwxr-xr-x 17 kw2018202065 kw2018202065 4096 Mar 15 00:13 ..
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$ chmod u-w,g-w,o-r hello.txt
kw2018202065@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-r--r--r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$ chmod 644 hello.txt
kw2018202065@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-r--r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$
```

chmod명령어는 파일에 대한 권한을 부여하는 명령어로 먼저 MODE를 설명하면 다음과 같다

대상	연산	권한
u: user(owner)	+ (추가)	r: read
g: group	- (제거)	w: write
o: other	= (할당)	x: execution
a: all		

또한 OCTAL-MODE를 지원하여 8진수로 나타낼 수 있으며 차례로 user(owner), group, other를 나타내고 1:execute, 2:write, 4:read를 나타낸다.

위의 사진으로 사용을 설명하면 다음과 같은데

먼저 hello.txt파일의 권한은 user에 rw권한 group에 rw권한 other에 r권한이 있었지만

chmod u-w,f-w,o-r hello.txt를 함으로써 user에 w제외 r권한 group에 w제외 r권한 group에 권한 없음으로 변경되었다. 또한 다시 chmod 664 hello.txt를 통하여 user에 6(w+r), group에 4(r) other에 4(r)을 줌으로써 -rw-r--r--권한으로 변경됨을 알 수 있다.

## -mkdir 명령어

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$ mkdir sp_lecture
kw2018202065@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 02:12 sp_lecture
kw2018202065@ubuntu:~/work$
```

mkdir명령어는 디렉토리 생성명령어로 위의 사진으로 설명하면 다음과 같다.

ls -l로 work디렉토리내의 파일을 출력하면 다음과 같은데 이후 mkdir sp\_lecture를 통해 work 디렉토리내에 sp\_lecture디렉토리를 생성했다.

## -rmdir 명령어

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 02:12 sp_lecture
kw2018202065@ubuntu:~/work$ rmdir sp_lecture
kw2018202065@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$
```

rmdir명령어는 디렉토리 삭제명령어로 work디렉토리내의 sp\_lecture디렉토리를 rmdir sp\_lecture를 통해 지워짐을 알 수 있다.

## -rm 명령어

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 02:18 LINUX
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$ rm -r LINUX
kw2018202065@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$
```

rm명령어는 파일 혹은 디렉토리 삭제 명령어로서 위의 사진을 참고하면 다음과 같다.

보통 파일을 지울때는 rm 파일명을 하면 되지만 디렉토리를 삭제할 시 rm -r옵션을 사용하여 디렉토리내의 파일을 재귀적으로 지워 삭제하게 된다. 위에서 보면 rm -r LINUX를 통해 LINUX디렉토리가 삭제됨을 알 수 있다. 또한 잘못 사용해서 삭제되는 것을 방지하기 위해 삭제할 때 정말로 삭제할것인지 되물어주는 rm -i옵션도 사용할 수 있다.



-copy 명령어

copy명령어는 파일 혹은 디렉토리를 복사하는 명령어로 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls -l
total 20
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$ cp hello.txt hello_copy.txt
kw2018202065@ubuntu:~/work$ ls -l
total 24
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 07:22 hello_copy.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
```

먼저 파일을 복사하는 예시로 cp hello.txt hello\_copy.txt를 통해 hello.txt파일을 hello\_copy.txt를 새로 만들어 복사하였다.(cp source target(file or directory)형식을 가진다.)

```
kw2018202065@ubuntu:~/work$ cp SP_lab/* .
kw2018202065@ubuntu:~/work$ ls -l
total 32
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileA.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileC.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 07:22 hello_copy.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 00:13 SP_lab
kw2018202065@ubuntu:~/work$
```

cp SP\_lab/\* .를 통해 sp\_lab디렉토리에 있는 모든파일 fileA.txt, fileB.txt를 복사하여 .을 통해 현재 폴더에 복사 붙여넣기 하였다.

-move명령어

move명령어는 파일들을 다른 디렉토리로 옮기거나 이름을 바꿔주는 역할을 하며 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls
ex file1.txt file2.txt file3.txt fileA.txt fileC.txt hello_copy.txt hello.txt SP_lab
kw2018202065@ubuntu:~/work$ mv hello_copy.txt /home/kw2018202065/work/ex
kw2018202065@ubuntu:~/work$ ls
ex file1.txt file2.txt file3.txt fileA.txt fileC.txt hello.txt SP_lab
kw2018202065@ubuntu:~/work$ cd ex
kw2018202065@ubuntu:~/work/ex$ ls
hello_copy.txt
kw2018202065@ubuntu:~/work/ex$ cd ..
kw2018202065@ubuntu:~/work$ mv ex LINUX
kw2018202065@ubuntu:~/work$ ls
file1.txt file2.txt file3.txt fileA.txt fileC.txt hello.txt LINUX SP_lab
kw2018202065@ubuntu:~/work$
```

먼저 형식은 move 파일 타겟 형식을 가지며 위의 사진에서는

mv hello\_copy.txt /home/kw2018202065/work/ex를 사용하여 hello\_copy.txt를 절대경로 /home/kw2018202065/work/ex에 옮긴 후 ls를 통해 이를 확인하였다.

또한 다시 work 디렉토리에서 mv ex LINUX를 통해 ex디렉토리를 LINUX디렉토리로 이름을 바꾼 것을 확인할 수 있다.

- ln 명령어

ln명령어는 파일 간의 링크를 만드는 명령어로서 윈도우의 바로가기 같은 파일을 만드는 명령어이다. ln명령어에는 Hard Link의 개념과 Symbolic Link의 개념이 있는데 아래에서 이 두 가지 개념을 설명하겠다.

### 1. Hard Link 경우

하드링크같은 경우에는 원본파일이 가리키는 inode를 하드링크된 파일도 공유하는 형태로 이루어지는데 이는 아래의 사진을 보면 알 수 있다.

아래는 하드링크를 사용하여 몇 가지 예시를 든 사진이다.

```
kw2018202065@ubuntu: ~/work/SP_lab
kw2018202065@ubuntu:~/work/SP_lab$ ls
fileA.txt fileC.txt
kw2018202065@ubuntu:~/work/SP_lab$ cat fileA.txt
This is file A
kw2018202065@ubuntu:~/work/SP_lab$ ln fileA.txt fileB.txt
kw2018202065@ubuntu:~/work/SP_lab$ ls
fileA.txt fileB.txt fileC.txt
kw2018202065@ubuntu:~/work/SP_lab$ cat fileB.txt
This is file A
kw2018202065@ubuntu:~/work/SP_lab$ vi fileB.txt
kw2018202065@ubuntu:~/work/SP_lab$ cat fileA.txt
This is file A
kw2018202065@ubuntu:~/work/SP_lab$ cat fileB.txt
This is file B after the change.
```

먼저 현재 디렉토리 안에 fileA.txt를 cat을 이용하여 출력하면 This is file A라는 문구가 나오고 ln fileA.txt fileB.txt (ln [소스] [타겟] 형태)를 통해 fileB.txt를 생성과 동시에 fileA.txt와 링크를 시켰다. 후에 cat을 사용하여 fileB.txt를 실행해보면 This is fileA라는 문구가 나오게 된다. 후에 Vi 편집기를 통해 fileB.txt안의 문구를 This is file B after change.라는 문구로 바꾸었는데 cat을 통해 fileA.txt를 확인한 결과 This is file B after change.라는 문구로 나오게 된다. 이는 하드링크를 통해 fileA.txt와 fileB.txt가 같은 inode 데이터 블록을 가리키기 때문이다.

### 2. Symbolic Link 경우

심볼릭 링크에서는 원본파일과 링크된 파일이 별개의 파일로 서로 다른 inode를 가리키게 되고 링크된 파일은 원본파일을 이름을 포함하는 포인터가 된다. 사용은 아래와 같다.

아래는 심볼릭링크를 사용하여 몇가지 예시를 든 사진이다.

```
kw2018202065@ubuntu: ~/work/SP_lab
kw2018202065@ubuntu:~/work/SP_lab$ cat fileC.txt
This is file C
kw2018202065@ubuntu:~/work/SP_lab$ ln -s fileC.txt fileE.txt
kw2018202065@ubuntu:~/work/SP_lab$ ls -l
total 12
-rw-rw-r-- 2 kw2018202065 kw2018202065 33 Mar 15 08:48 fileA.txt
-rw-rw-r-- 2 kw2018202065 kw2018202065 33 Mar 15 08:48 fileB.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 08:13 fileC.txt
lrwxrwxrwx 1 kw2018202065 kw2018202065 9 Mar 15 09:06 fileE.txt -> fileC.txt
kw2018202065@ubuntu:~/work/SP_lab$ cat fileE.txt
This is file C
kw2018202065@ubuntu:~/work/SP_lab$ rm fileC.txt
kw2018202065@ubuntu:~/work/SP_lab$ cat fileE.txt
cat: fileE.txt: No such file or directory
kw2018202065@ubuntu:~/work/SP_lab$
```

먼저 현재디렉토리내에서 cat 명령어를 통해 fileC.txt를 출력한 결과 This is file C라는 문구가 나온다. 이후 ln -s 옵션(Symbolic Link)을 사용하여 ln -s fileC.txt fileE.txt를 했다. 그 후 ls -l을 통해 디렉토리의 파일들을 자세히 확인하면 fileE.txt가 fileC.txt를 가리킨다고 알 수 있고 cat을 통해 fileE.txt를 출력하면 his is file C 즉 fileC.txt와 같은 문구가 출력된다. 후에 rm명령어로 fileC.txt를 지우고 fileE.txt를 출력하면 찾을 수 없다는 오류 메시지를 출력하는데 이는 fileA.txt와 fileB.txt가 서로 다른 inode 데이터 블록을 가리키며 링크된 파일은 원본파일의 포인터를 가리키게 되지만 원본파일이 삭제되어 나타나는 오류이다.

Hard Link 와 Symbolic Link의 차이점을 표로 나타내면 아래와 같다.

Hard Link	Symbolic Link
파일만 링크가능 디렉토리 링크불가능	파일 또는 디렉토리 링크 가능
존재하지 않는 파일에 대한 링크 불가능	존재하지 않는 파일에 대한 링크 가능
연결되어 있는 파일인지 알기 어려움	연결되어 있는 파일을 찾기 용이(ls 명령어 이용 등)
다른 파일 시스템간의 하드링크불가능	NFS나 RFS상에서도 심볼릭 링크 사용가능
원본 파일과 I-node같은	원본 파일과 I-node다름

-touch 명령어

touch 명령어는 빈 파일을 생성하거나 존재하는 파일의 타임스탬프를 변경하는 용도로 사용된다.

아래는 touch명령어를 사용한 사진이다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls
file1.txt file2.txt file3.txt fileA.txt fileC.txt hello.txt LINUX SP_lab
kw2018202065@ubuntu:~/work$ touch empty.txt
kw2018202065@ubuntu:~/work$ ls
empty.txt file1.txt file2.txt file3.txt fileA.txt fileC.txt hello.txt LINUX SP_lab
kw2018202065@ubuntu:~/work$ ls -l
total 32
-rw-rw-r-- 1 kw2018202065 kw2018202065  0 Mar 15 09:42 empty.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileA.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileC.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 07:30 LINUX
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 09:12 SP_lab
```

touch empty.txt를 통해 empty.txt를 만들어 주었고 ls -l를 통해 수정시간을 확인하면 09:42으로 나오게 된다.

```
kw2018202065@ubuntu:~/work$ touch empty.txt
kw2018202065@ubuntu:~/work$ ls -l
total 32
-rw-rw-r-- 1 kw2018202065 kw2018202065  0 Mar 15 09:46 empty.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileA.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileC.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 07:30 LINUX
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 09:12 SP_lab
kw2018202065@ubuntu:~/work$
```

이후 다시 touch empty.txt를 하면 존재하는 파일의 타임스탬프가 변경됨을 알 수 있다.

-ps 명령어

ps명령어는 사용중인 프로세스의 정보를 알 수 있도록 하는 명령어이고 사용은 아래와 같다.

아래는 ps명령어를 -e, -f옵션을 사용하여 출력한 사진이다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1        0  0  02:37 ?        00:00:03 /sbin/init auto noprompt
root           2        0  0  02:37 ?        00:00:00 [kthreadd]
root           4        2  0  02:37 ?        00:00:00 [kworker/0:0H]
root           6        2  0  02:37 ?        00:00:00 [mm_percpu_wq]
root           7        2  0  02:37 ?        00:00:00 [ksoftirqd/0]
root           8        2  0  02:37 ?        00:00:02 [rcu_sched]
root           9        2  0  02:37 ?        00:00:00 [rcu_bh]
root          10        2  0  02:37 ?        00:00:00 [migration/0]
root          11        2  0  02:37 ?        00:00:00 [watchdog/0]
root          12        2  0  02:37 ?        00:00:00 [cpuhp/0]
root          13        2  0  02:37 ?        00:00:00 [cpuhp/1]
root          14        2  0  02:37 ?        00:00:00 [watchdog/1]
root          15        2  0  02:37 ?        00:00:00 [migration/1]
root          16        2  0  02:37 ?        00:00:00 [ksoftirqd/1]
root          18        2  0  02:37 ?        00:00:00 [kworker/1:0H]
root          19        2  0  02:37 ?        00:00:00 [cpuhp/2]
root          20        2  0  02:37 ?        00:00:00 [watchdog/2]
root          21        2  0  02:37 ?        00:00:00 [migration/2]
root          22        2  0  02:37 ?        00:00:02 [ksoftirqd/2]
root          24        2  0  02:37 ?        00:00:00 [kworker/2:0H]
root          25        2  0  02:37 ?        00:00:00 [kdevtmpfs]
root          26        2  0  02:37 ?        00:00:00 [netns]
root          27        2  0  02:37 ?        00:00:00 [rcu_tasks_kthre]
root          28        2  0  02:37 ?        00:00:00 [kauditd]
root          32        2  0  02:37 ?        00:00:00 [khungtaskd]
root          33        2  0  02:37 ?        00:00:00 [oom_reaper]
root          34        2  0  02:37 ?        00:00:00 [writeback]
root          35        2  0  02:37 ?        00:00:00 [kcompactd0]
root          36        2  0  02:37 ?        00:00:00 [ksmd]
root          37        2  0  02:37 ?        00:00:00 [khugepaged]
root          38        2  0  02:37 ?        00:00:00 [crypto]
root          39        2  0  02:37 ?        00:00:00 [kintegrityd]
root          40        2  0  02:37 ?        00:00:00 [kblockd]
root          41        2  0  02:37 ?        00:00:00 [ata_sff]
root          42        2  0  02:37 ?        00:00:00 [md]
root          43        2  0  02:37 ?        00:00:00 [edac-poller]
root          44        2  0  02:37 ?        00:00:00 [devfreq_wq]
root          45        2  0  02:37 ?        00:00:00 [watchdogd]
```

ps에 -e옵션(모든 프로세스를 출력), -f(full format으로 출력)을 사용하여 현재 사용중인 프로세스를 출력했다.

-exit 명령어

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ sudo apt-get install csh
[sudo] password for kw2018202065:
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 499 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 1s (137 kB/s)
Selecting previously unselected package csh.
(Reading database ... 177265 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) ...
Setting up csh (20110502-2.1ubuntu1) ...
update-alternatives: using /bin/bsd-csh to provide /bin/csh (csh) in auto mode
kw2018202065@ubuntu:~/work$ ps
  PID TTY          TIME CMD
 2229 pts/4    00:00:00 bash
 7321 pts/4    00:00:00 ps
kw2018202065@ubuntu:~/work$ csh
% ps
  PID TTY          TIME CMD
 2229 pts/4    00:00:00 bash
 7322 pts/4    00:00:00 csh
 7323 pts/4    00:00:00 ps
% exit
kw2018202065@ubuntu:~/work$ ps
  PID TTY          TIME CMD
 2229 pts/4    00:00:00 bash
 7324 pts/4    00:00:00 ps
kw2018202065@ubuntu:~/work$
```

먼저 sudo apt-get install csh를 통해 csh를 설치하고 ps를 통해 현재 사용중인 프로세스를



확인한다. 이때는 csh가 없다. 이후 csh를 치면 csh셸로 넘어가게 되고 이때 ps를 치게 되면 csh를 확인할 수 있다. 그 후 exit을 하면 csh셸을 종료하게 되고 ps를 통해 확인 가능하다. 즉 셸을 종료하는 명령어이다.

#### -kill 명령어

kill명령어는 프로세스에 특정한 signal을 보내는 명령어로서 일반적으로 종료되지 않는 프로세스를 종료 시킬 때 많이 사용한다.

아래의 사진은 kill명령어의 사용예시이다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ kill -l
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS     8) SIGFPE     9) SIGKILL    10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM    15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD   18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU   23) SIGURG     24) SIGXCPU   25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF   28) SIGWINCH   29) SIGIO      30) SIGPWR
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
53) SIGRTMAX-1 64) SIGRTMAX
kw2018202065@ubuntu:~/work$ ps -e | tail
6605 ?      00:00:00 kworker/2:1
7337 ?      00:00:00 kworker/1:1
7387 ?      00:00:02 kworker/u256:3
7401 ?      00:00:03 kworker/u256:4
7412 pts/4    00:00:04 yes
7431 ?      00:00:00 kworker/1:2
7438 ?      00:00:00 kworker/u256:0
7442 ?      00:00:00 kworker/u256:1
7455 pts/4    00:00:00 ps
7456 pts/4    00:00:00 tail
kw2018202065@ubuntu:~/work$ kill -9 7412
kw2018202065@ubuntu:~/work$ ps -e | tail
6415 ?      00:00:00 kworker/0:0
6605 ?      00:00:00 kworker/2:1
7337 ?      00:00:00 kworker/1:1
7387 ?      00:00:02 kworker/u256:3
7401 ?      00:00:03 kworker/u256:4
7431 ?      00:00:00 kworker/1:2
7438 ?      00:00:00 kworker/u256:0
7442 ?      00:00:00 kworker/u256:1
7457 pts/4    00:00:00 ps
7458 pts/4    00:00:00 tail
[1]+  Killed                  yes my name
kw2018202065@ubuntu:~/work$
```

kill -l를 통해 보낼 수 있는 시그널의 종류를 알 수 있다. ps -e | tail 즉 모든 프로세스를 출력하는데 이전 명령어의 output을 다음명령어의 input으로 연결해 주는 파이프라인을 통해 끝에서 10개만 출력하면 7412번 프로세스인 yes를 볼 수 있다. 이 프로세스를 -9(강제 종료 시그널)을 사용하여 kill -9 7412를 하고 다시 ps -e | tail를 하면 yes프로세스가 강제 종료됨을 알 수 있다.

#### -passwd 명령어

```
kw2018202065@ubuntu: ~
kw2018202065@ubuntu:~$ passwd
Changing password for kw2018202065.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
You must choose a longer password
Enter new UNIX password: 
```

user의 비밀번호를 변경하는 명령어이다. passwd를 입력하면 현재비밀번호를 입력하라는 문구가 나오고 입력을 하면 새로운 비밀번호를 입력하는 문구가 나오는 형식이다.



- uname 명령어

시스템 정보를 출력하는 명령어로 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~  
kw2018202065@ubuntu:~$ uname  
Linux  
kw2018202065@ubuntu:~$ uname -r  
4.15.0-29-generic  
kw2018202065@ubuntu:~$ uname -m  
x86_64  
kw2018202065@ubuntu:~$ uname -a  
Linux ubuntu 4.15.0-29-generic #31~16.04.1-Ubuntu SMP Wed Jul 18 08:54:04 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux  
kw2018202065@ubuntu:~$
```

uname을 하면 LINUX시스템이라는 것을 알 수 있다.

uname -r을 하면 커널 버전을 알 수 있다.

uname -m을 하면 사용자의 machine hardware의 이름이 나오는 것을 알 수 있다.

uname -a를 하면 모든 정보를 출력해준다.

- wc 명령어

파일에 대한 newline, word, byte 개수를 출력해준다. 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~/work  
kw2018202065@ubuntu:~/work$ cat hello.txt  
hello world  
My Name is N~~~  
How are you?  
kw2018202065@ubuntu:~/work$ wc hello.txt  
 3  9 41 hello.txt  
kw2018202065@ubuntu:~/work$ wc -c hello.txt  
41 hello.txt  
kw2018202065@ubuntu:~/work$ wc -m hello.txt  
41 hello.txt  
kw2018202065@ubuntu:~/work$ wc -l hello.txt  
3 hello.txt  
kw2018202065@ubuntu:~/work$ wc -L hello.txt  
15 hello.txt  
kw2018202065@ubuntu:~/work$ wc -w hello.txt  
9 hello.txt
```

wc hello.txt를 통해 차례로 newline, word, byte 개수를 알 수 있다.

wc -c hello.txt를 통해 바이트 수를 알 수 있다.

wc -m hello.txt를 통해 문자 수를 알 수 있다.

wc -l hello.txt를 통해 줄수를 알 수 있다.

wc -L hello.txt를 통해 가장 긴 줄의 길이를 알 수 있다.

wc -w hello.txt를 통해 단어 수를 알 수 있다.

- echo 명령어

echo명령어는 string을 출력하는 명령어로 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~/work  
kw2018202065@ubuntu:~/work$ echo helloworld  
helloworld  
kw2018202065@ubuntu:~/work$ echo $HOME  
/home/kw2018202065  
kw2018202065@ubuntu:~/work$ echo ~  
/home/kw2018202065  
kw2018202065@ubuntu:~/work$
```

echo helloworld를 통해 helloworld문구를 출력함을 알 수 있고 홈 디렉토리의 경로를 담은 \$HOME과 ~를 echo \$HOME과 echo ~ 명령어를 통해 홈 디렉토리의 경로를 출력하였다.

#### - alias 명령어

alias명령어는 단어를 다른 문자열로 대체하는 것을 가능하게 한다. 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ alias myls='ls -al'
kw2018202065@ubuntu:~/work$ myls
total 40
drwxrwxr-x 4 kw2018202065 kw2018202065 4096 Mar 15 09:42 .
drwxr-xr-x 17 kw2018202065 kw2018202065 4096 Mar 15 10:30 ..
-rw-rw-r-- 1 kw2018202065 kw2018202065 0 Mar 15 09:46 empty.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file1.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 21 Mar 15 00:13 file2.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 2001 Mar 15 00:13 file3.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileA.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 15 Mar 15 07:23 fileC.txt
-rw-rw-r-- 1 kw2018202065 kw2018202065 41 Mar 15 00:13 hello.txt
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 07:30 LINUX
drwxrwxr-x 2 kw2018202065 kw2018202065 4096 Mar 15 09:12 SP_lab
kw2018202065@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$//'\`)"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
alias myls='ls -al'
kw2018202065@ubuntu:~/work$
```

alias myls='ls -al'를 입력하여 myls의 단어는 ls -al의 문자열로 대체했고 이는 디렉토리의 모든 파일을 자세히 출력하라는 명령어이기 때문에 myls를 입력했을 때 ls -al의 역할을 하는 것을 알 수 있다. 또한 alias만 입력을 하면 앞서 대체한 myls와 비롯하여 alias된 모든 단어들을 출력해준다.

#### - grep 명령어

grep 명령어는 특정 파일에서 패턴을 이용한 지정한 문자열이나 정규표현식을 포함한 행을 출력해주는 명령어이다.(특정 파일이 지정되지 않은 경우 표준 입력이 사용된다.)

grep 명령어의 사용은 아래와 같다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ grep hello hello.txt
hello world
kw2018202065@ubuntu:~/work$ grep hello
hello.txt
hello.txt
```

grep hello hello.txt를 통해 hello를 사용한 행을 hello.txt에서 출력을 해준다.

또한 grep hello를 통해 특정 파일을 지정하지 않은 경우 표준 입력이 사용되고 여기에 hello.txt를 입력하면 hello를 사용한 행을 hello.txt에서 출력을 해준다.


- Vi editor 사용 및 설명

```
kw2018202065@ubuntu: ~  
2018202065  
PARKCHEOLJUN  
helloworld  
~
```

터미널에서 vi 입력 후 Vi editor를 켜다. 명령모드에서 a를 눌러 입력모드에 접속 후에 첫째 줄에 학번 2018202065를 입력하고 둘째줄에 이름 PARKCHEOLJUN을 입력하고 셋째줄에 helloworld를 입력하였다.

```
kw2018202065@ubuntu: ~  
2018202065  
helloworld  
PARKCHEOLJUN  
helloworld
```

esc를 눌러 명령모드로 돌아간후 커서를 제일 마지막줄 helloworld에 두고 yy를 입력해 helloworld행 전체를 복사한 후 학번이 있는 줄에 커서를 두고 p를 입력하여 붙여넣기를 한다.



The screenshot shows a terminal window with a dark background and light-colored text. The window title bar at the top reads "kw2018202065@ubuntu: ~". The terminal content consists of a list of four items, each on a new line and preceded by a number in a light blue font:

- 1 2018202065
- 2 helloworld
- 3 PARKCHEOLJUN
- 4 helloword

The cursor is positioned at the end of the fourth line, after the letter 'd' in "helloword". The terminal window is open on a desktop with a dark background and a grid of application icons on the left side.

다시 명령모드에서 :를 입력하고 set number 즉 :set number를 입력하면 편집기의 라인이 표시되게 된다.



이후 make를 사용하기 위해 Makefile파일을 만들었고 이는 아래와 같다.

```
kw2018202065@ubuntu: ~/work
OBJS = kw_hello_c.c
CC = gcc
EXEC = hello

all: $(OBJS)
    $(CC) -o $(EXEC) $^

clean:
    rm -rf $(EXEC)
```

6,1-8 All

먼저 변수 OBJS, CC, EXEC를 만들고 각각에 kw\_hello\_c.c와 gcc, hello를 넣어주고 타겟에 all 타겟 명에 변수를 사용해서 \$(OBJS) 즉 all : kw\_hello\_c.c를 해준 뒤 \$(cc) - o \$(EXEC) \$^ 즉 gcc -o hello kw\_hello\_c.c를 해준다. clean 부분은 더미타겟으로 의존성 파일들을 정의하지 않아서 파일을 생성하지 않는다. 대신, make로 명령을 실행 시 생성되는 Target파일들을 제거해주는 용도로 사용된다. make를 수행 전 안전하게 파일들을 제거하고 다시 새롭게 하기 위한 용도로 사용되고 명령은 'make clean' 을 이용한다. 이제 이 파일을 이용하면 아래와 같다.

```
kw2018202065@ubuntu: ~/work
kw2018202065@ubuntu:~/work$ ls
empty.txt file1.txt file2.txt file3.txt fileA.txt fileC.txt hello.txt kw_hello_c.c LINUX Makefile SP_lab
kw2018202065@ubuntu:~/work$ make
gcc -o hello kw_hello_c.c
kw2018202065@ubuntu:~/work$ ls
empty.txt file1.txt file2.txt file3.txt fileA.txt fileC.txt hello hello.txt kw_hello_c.c LINUX Makefile SP_lab
kw2018202065@ubuntu:~/work$ ./hello
2018202065 PARKCHEOLJUN
kw2018202065@ubuntu:~/work$
```

먼저 Makefile이 있는 디렉토리내에서 ls를 해보면 kw\_hello\_c.c와 Makefile이 있는 것을 알 수 있다. 그리고 make명령어를 사용하면 전에 Makefile에서 만들어 준 매크로가 실행 되어 컴파일 되고 다시 ls를 입력하면 hello라는 실행파일이 생성됨을 알 수 있다. 그 후 hello파일을 실행하면 위에서 작성한 kw\_hello\_c.c의 코드내용에 맞게 학번과 이름이 출력된다.



-고찰

이번 과제를 진행하면서 GUI환경에 익숙한 나에게 CLI환경인 리눅스에서 콘솔모드로 효율적인 작업 수행을 하기위해선 과제를 통해 리눅스 명령어 습득이 가장 중요함을 알게되었다. 또한 서버나 임베디드 소프트웨어 등 리눅스를 사용하는 CLI환경에서 텍스트 기반 파일의 작업을 할 때 윈도우에서 흔하게 쓰는 텍스트 편집기를 사용할 수 없기에 vi 편집기를 이용해서 작성을 해야한다. 이를 위해서는 vi편집기의 사용을 원활하게 할 수 있어야 하고 이번 과제를 통해 vi 편집기의 활용을 익힐 수 있었다.