

1차 과제 – Linux 기초

1-1차: Ubuntu Installation

시스템 프로그래밍 실습

제출일: 3월 29일 금요일

분 반: 화요일

담당 교수: 신영주

학 번: 2015722025

학 과: 컴퓨터정보공학부

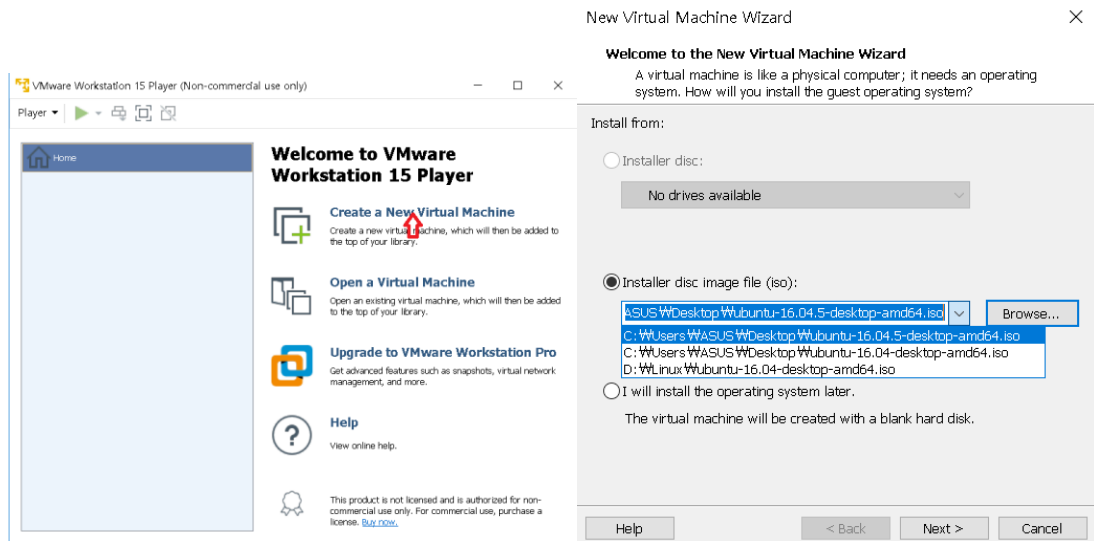
이 름: 정용훈

1. Introduction

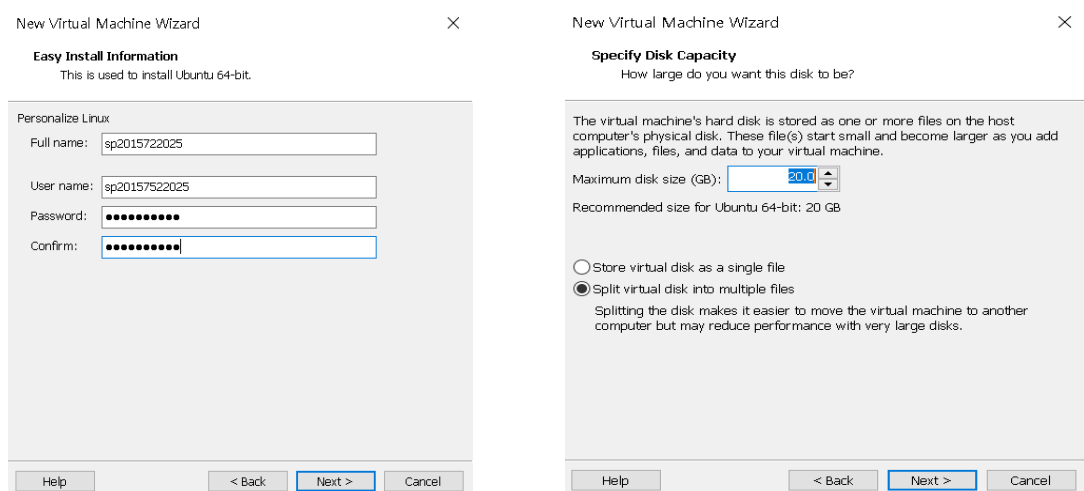
앞으로 시스템 프로그래밍을 배우면서 사용하게 될 Linux OS의 배포판 중 하나인 Ubuntu의 설치과정을 정리하는데 목적을 두고 있다.

2. Result

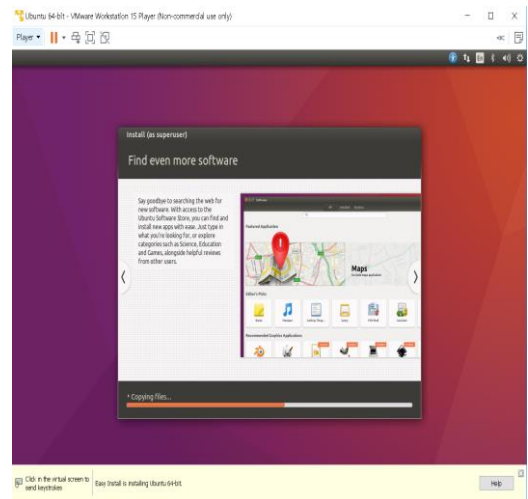
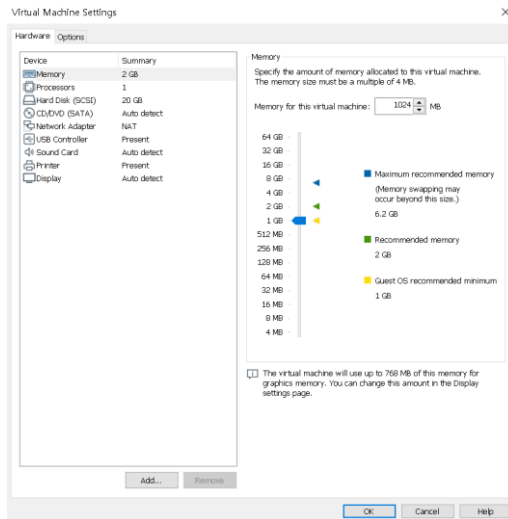
다음은 설치과정을 나타낸 것으로 해당 과정을 따라가면 쉽게 Ubuntu를 설치 할 수 있다. 앞선 과제의 조건인 Workstation의 설치 과정은 생략한다.



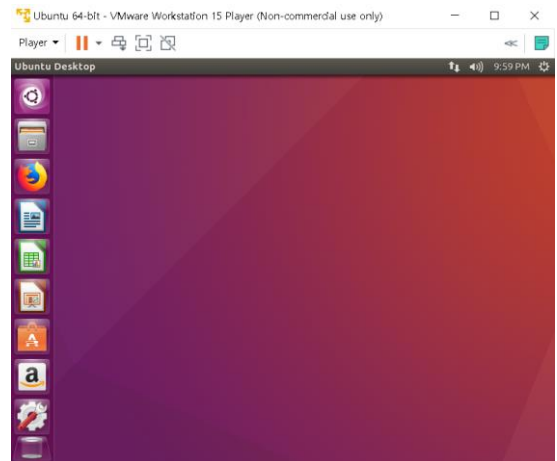
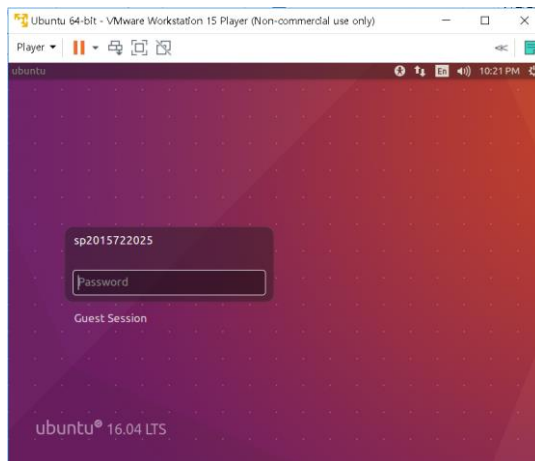
다음 그림처럼 Workstation을 설치한 후 Virtual machine을 생성할 수 있다. 해당 버튼을 클릭한 후 실습시간에 배포받은 iso파일을 불러와 Next버튼을 눌러준다.



다음으로 Virtual machine에서 사용할 사용자이름과 password를 설정한 후 machine의 각종 설정을 해준다. 오른쪽 그림은 machine의 최대 disk size를 20GB로 설정해준 모습이 다.



다음은 제공된 pdf처럼 ram을 1GB로 설정해준 후 Finish버튼을 눌러주면 오른쪽 화면처럼 설치를 진행한다.



설치가 완료된 후 사용자를 선택하여 password를 입력하고 로그인할 수 있는 화면이 나오며 오른쪽은 로그인을 성공한 모습이다. 앞으로는 terminal을 통해 각종 실습을 하게 된다.

3. Reference

강의 자료와 실습시간에 배포된 자료를 통하여 파일을 충분히 다운로드 받을 수 2018년도 오픈소스 소프트웨어 설계 및 실습에서의 강의자료를 참고하여 설치했습니다.

1차 과제 – Linux 기초

1-2차: Usage of Linux Commands

시스템 프로그래밍 실습

제출일: 3월 29일 금요일

분 반: 화요일

담당 교수: 신영주

학 번: 2015722025

학 과: 컴퓨터정보공학부

이 름: 정용훈

4. Introduction

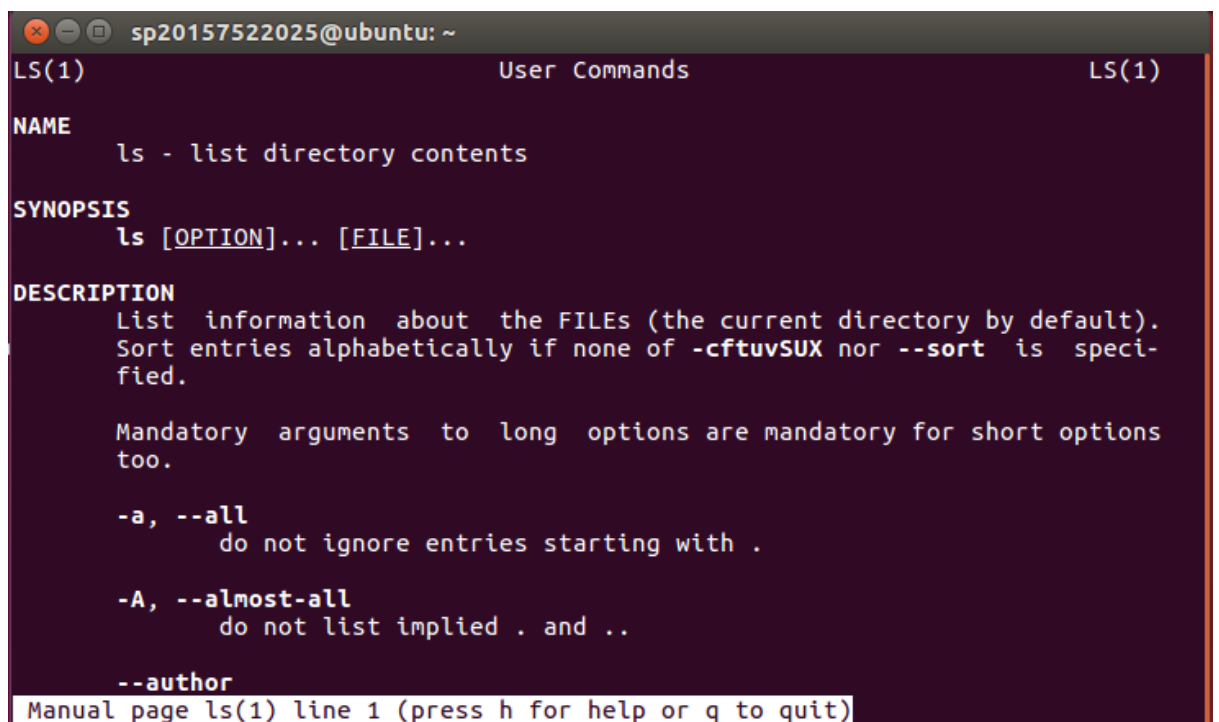
3월 12일 실습시간을 통하여 배운 여러 가지 기본적인 Linux명령어를 이해하고 익히며, 해당 내용을 정리하는데 목적을 둔다.

5. Result

(1) Man

해당 명령어는 다른 각 명령어의 매뉴얼, 즉 사용 방법이나 명칭을 알고자 할 때 사용하는 명령어로 실행하면 다음과 같은 결과를 볼 수 있다.

```
sp20157522025@ubuntu:~$ man ls
```



```
sp20157522025@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

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

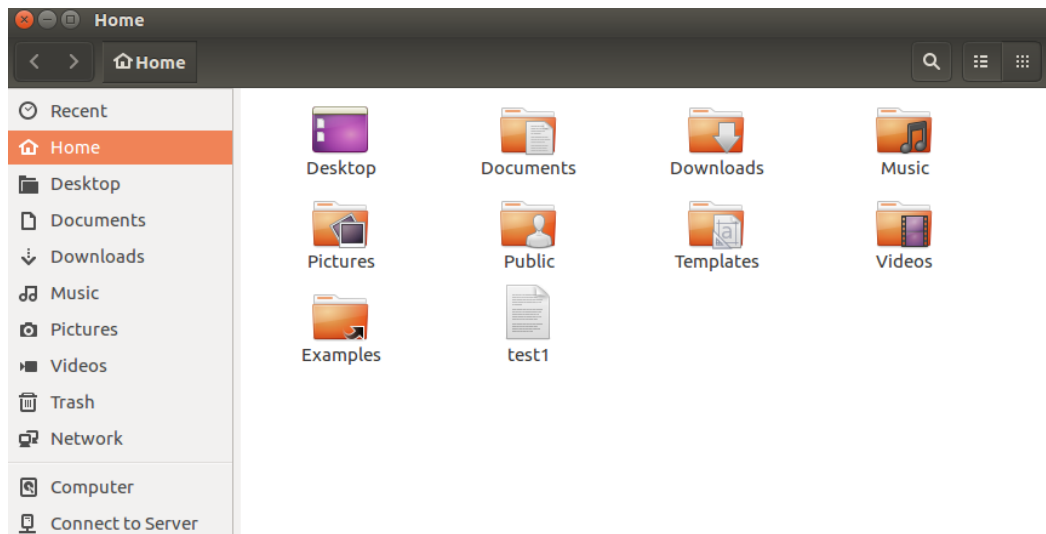
ls명령어에 대한 명칭과 사용 방법 등이 설명 되어있는 것을 확인할 수 있다. 또한 옵션 명령어를 통한 다른 출력이 가능한데 -k는 키워드를 통하여 명령어를 찾아주며, -a는 모든 매뉴얼을 원할 때 사용하게 된다.

(2) ls

ls는 List directory contents라는 뜻으로 앞으로 가장 많이 사용하게 될 명령어라고 해도 과언이 아닌 명령이다. 해당 명령의 역할은 현재 위치에서의 파일의 목록을 알려주는 기능을 수행한다.

```
sp20157522025@ubuntu:~$ ls
```

```
Desktop  Downloads  Music  Public  test1
Documents examples.desktop Pictures Templates Videos
```



다음과 같이 home에 있는 파일들의 목록을 나열해준다. ls명령 또한 옵션이 존재하는데 옵션들은 -a, -F, -l이 존재 한다. 각각의 옵션은 다음과 같은 뜻이다.

-a : hidden file을 포함한 모든 파일 출력

-F : 파일을 종류 표시(/는 디렉토리, *는 실행파일)

-l : 파일의 정보를 자세하게 출력

해당 명령어들의 실행결과는 다음과 같다.

```
sp20157522025@ubuntu:~$ ls -a
```

```
. Desktop .ICEauthority test1
.. .dmrc .local tset folder
.bash_history Documents Music Videos
.bash_logout Downloads Pictures .Xauthority
.bashrc examples.desktop .profile .xsession-errors
.cache .gconf Public .xsession-errors.old
.config .gnupg Templates
```

```
sp20157522025@ubuntu:~$ ls -F
```

```
Desktop/ Downloads/ Music/ Public/ test1 Videos/
Documents/ examples.desktop Pictures/ Templates/ tset folder/
```

```
sp20157522025@ubuntu:~$ ls -l
```

```
total 48
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Desktop
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Documents
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Downloads
-rw-r--r-- 1 sp20157522025 sp20157522025 8980 Mar 15 19:55 examples.desktop
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Music
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Pictures
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Public
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Templates
-rw-rw-r-- 1 sp20157522025 sp20157522025 0 Mar 15 22:37 test1
drwxrwxr-x 2 sp20157522025 sp20157522025 4096 Mar 15 22:47 tset folder
drwxr-xr-x 2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Videos
```

(3) pwd

해당 명령어는 현재 작업이 진행되는 directory를 표시해주는 명령어다. 해당 명령어를 사용하면 다음과 같은 결과를 확인할 수 있다.

```
sp20157522025@ubuntu:~$ pwd
```

```
/home/sp20157522025
```

다음은 작업 파일을 이동한 후 해당 명령을 사용했을 때의 결과 화면이다.

```
sp20157522025@ubuntu:~/testfolder$ pwd
```

```
/home/sp20157522025/testfolder
```

결과와 같이 사용자가 현재 작업하는 환경의 위치를 알려줄 때 사용한다.

(4) Cd

cd또한 ls와 마찬가지로 굉장히 많이 쓰이는 명령어 중 하나이다. Cd의 뜻은 change the current directory로 작업하는 directory를 바꾸는데 사용하게 된다.

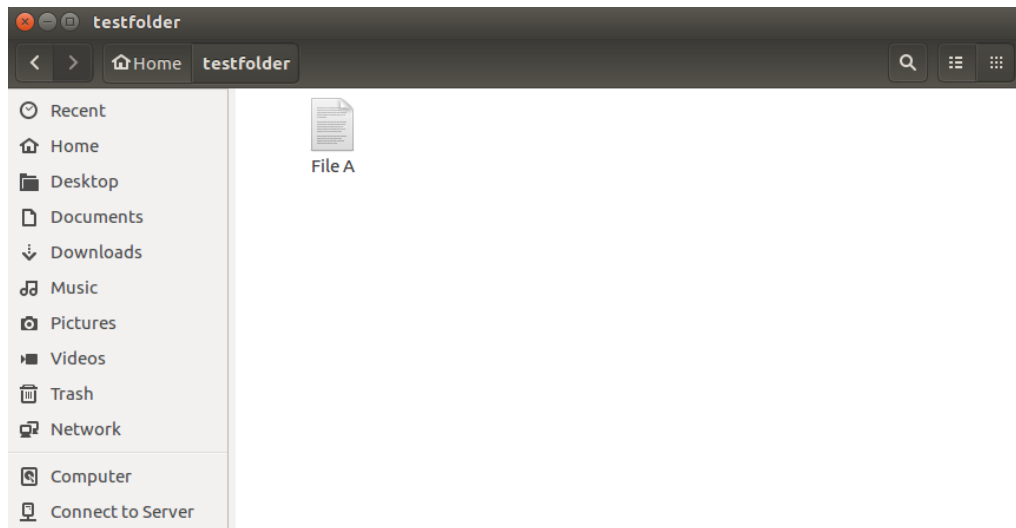
```
sp20157522025@ubuntu:~$ cd testfolder
```

```
sp20157522025@ubuntu:~/testfolder$
```

작업 환경이 home에서 testfolder로 옮겨간 상태다. 현재 상태에서 ls명령을 사용하면 다음과 같이 나온다.

```
sp20157522025@ubuntu:~/testfolder$ ls
```

File A



Testfolder에 File A라는 파일 하나만 있으므로 결과가 정상적으로 나왔다는 것을 쉽게 알 수 있다. 또한 cd명령에는 cd명령을 조금 더 간편하게 사용할 수 있는 Special filenames이 있다. 해당 명령은 ., ..으로 나타낼 수 있으면 해당 명령은 다음과 같은 뜻을 내포하고 있다.

. : (current directory)

.. : (parent directory)

해당 명령은 다음과 같이 사용할 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ cd .
```

```
sp20157522025@ubuntu:~/testfolder$
```

다음과 같이 현재 directory로 이동하는 명령어 이므로 작업환경이 변경되지 않는 모습이다.

```
sp20157522025@ubuntu:~/testfolder/testfolder2/testfolder3$ cd ../../
```

```
sp20157522025@ubuntu:~/testfolder$
```

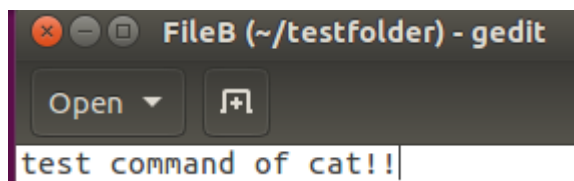
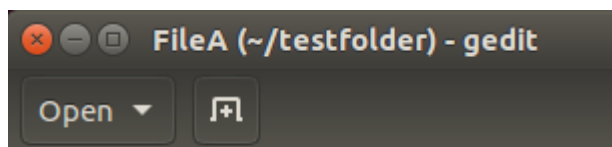
다음결과를 이전, 이전 directory로 이동해야 하므로 다시 testfolder로 이동한 모습을 확인 할 수 있다.

(5) Cat

Cat의 뜻은 concatenate files and print on the standard output으로 directory가 포함하고 있는 파일의 내용을 출력해주는 역할을 한다. 명령의 결과는 다음과 같이 나타낼 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ ls
FileA  FileB  testfolder2
```

```
sp20157522025@ubuntu:~/testfolder$ cat FileA
This is File A
sp20157522025@ubuntu:~/testfolder$ cat FileB
test command of cat!!
sp20157522025@ubuntu:~/testfolder$ cat FileA FileB
This is File A
test command of cat!!
```



결과가 잘 출력되는 것을 확인할 수 있다.

(6) Chomd

해당 명령어는 파일의 권한을 변경해주는 명령어로 change mode라는 의미를 가지고 있는 명령어다. Chmod명령은 다음과 같이 사용할 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ ls -al
total 20
drwxrwxr-x  3 sp20157522025 sp20157522025 4096 Mar 15 23:22 .
drwxr-xr-x 16 sp20157522025 sp20157522025 4096 Mar 15 22:57 ..
-rw-rw-r--  1 sp20157522025 sp20157522025   15 Mar 15 22:55 FileA
-rw-rw-r--  1 sp20157522025 sp20157522025   22 Mar 15 23:21 FileB
drwxrwxr-x  3 sp20157522025 sp20157522025 4096 Mar 15 23:09 testfolder2
```

우선 chmod명령을 사용하기 위해선 각 파일과 폴더의 권한이 어떻게 지정되어있는지 이해해야 하는 것이 먼저다. 위 이미지에서 보이는 각 directory의 권한은 앞에 drwxrwxr-x로 쓰여있는 부분에서 알 수 있으며, 앞에 쓰여있는 d는 directory폴더를 의미한다. 해당자리에는 d, l, - 이 올 수 있으며 각각의 뜻은 다음과 같다.

d : Directory, 폴더를 의미

l : link file, 링크 파일을 의미

- : 일반 정규파일을 의미

그 다음에 오는 drwxrwxr-x의 의미다. 다음 표시를 3가지 색으로 나누었는데 각 색은 파일을 접근하는 사용자가 3가지로 나뉘기 때문이다. 3가지로 나누면 빨간색은 user, 파란색은 group, 초록색은 others가 된다. 각각 rwx의 뜻은 다음과 같다.

r : 읽기권한 있음

w : 쓰기권한 있음

x : 실행권한 있음

- : 권한 없음

각각의 알파벳은 다음과 같은 뜻을 내포하고 있으며, 각 파일이나 디렉토리는 chmod 명령을 통하여 권한을 설정할 수 있다. 권한을 설정하는 결과는 다음과 같다.

```

sp20157522025@ubuntu:~/testfolder$ chmod u-w,g-w,o-r FileA
sp20157522025@ubuntu:~/testfolder$ ls -al
total 20
drwxrwxr-x 3 sp20157522025 sp20157522025 4096 Mar 15 23:22 .
drwxr-xr-x 16 sp20157522025 sp20157522025 4096 Mar 15 22:57 ..
-r--r----- 1 sp20157522025 sp20157522025 15 Mar 15 22:55 FileA
-rw-rw-r-- 1 sp20157522025 sp20157522025 22 Mar 15 23:21 FileB
drwxrwxr-x 3 sp20157522025 sp20157522025 4096 Mar 15 23:09 testfolder2

```

다음은 FileA의 권한을 제거하는데 사용한 명령어다. 결과와 같이 권한이 `-r--r-----`으로 바뀐 것을 볼 수 있다. 다시 권한을 주려면 아래와 같이 실행시키면 된다.

```

sp20157522025@ubuntu:~/testfolder$ chmod u+w,g+w,o+r FileA
sp20157522025@ubuntu:~/testfolder$ ls -al
total 20
drwxrwxr-x 3 sp20157522025 sp20157522025 4096 Mar 15 23:22 .
drwxr-xr-x 16 sp20157522025 sp20157522025 4096 Mar 15 22:57 ..
-rw-rw-r-- 1 sp20157522025 sp20157522025 15 Mar 15 22:55 FileA
-rw-rw-r-- 1 sp20157522025 sp20157522025 22 Mar 15 23:21 FileB
drwxrwxr-x 3 sp20157522025 sp20157522025 4096 Mar 15 23:09 testfolder2
sp20157522025@ubuntu:~/testfolder$ █

```

+를 통하여 다시 권한을 추가한 것을 확인할 수 있다. 또한 권한을 부여하는데 8진수를 사용하여 나타낼 수 있는데 아래와 같이 설명할 수 있다.

`rw- rw- r--`

110 110 100 >2진수

6 4 4 >8진수

위와 같이 권한을 0과 1로 나타낼 수 있으며 8진수를 통하여 명령을 다음과 같이 부여할 수 있다.

```

sp20157522025@ubuntu:~/testfolder$ chmod 777 FileA
sp20157522025@ubuntu:~/testfolder$ ls -al
total 20
drwxrwxr-x 3 sp20157522025 sp20157522025 4096 Mar 15 23:22 .
drwxr-xr-x 16 sp20157522025 sp20157522025 4096 Mar 15 22:57 ..
-rwxrwxrwx 1 sp20157522025 sp20157522025 15 Mar 15 22:55 FileA
-rw-rw-r-- 1 sp20157522025 sp20157522025 22 Mar 15 23:21 FileB
drwxrwxr-x 3 sp20157522025 sp20157522025 4096 Mar 15 23:09 testfolder2

```

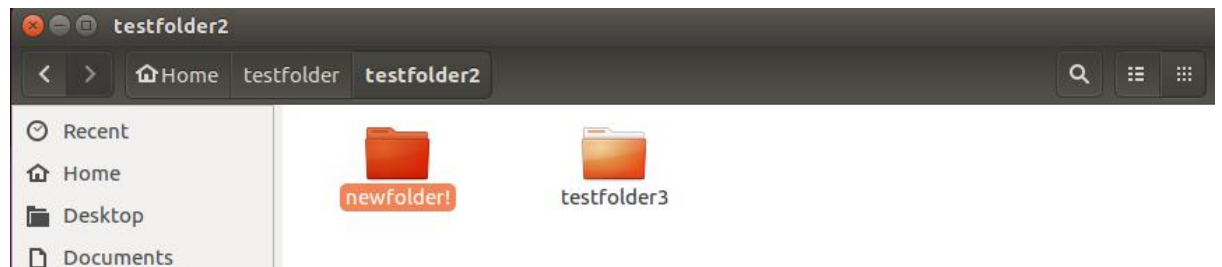
7은 2진수로 1 1 1로 나타내므로 모든 r, w, x의 권한을 모두 부여하는 것으로 777로 FileA에 명령을 주면 `-rwxrwxrwx`로 user, group, others에게 모든 권한이 부여되는 것을 확인할 수 있다.

(7) Mkdir

Mkdir명령은 make directories로 현재 작업하는 위치에서 directory을 생성하는 명령어다. 다음 명령의 결과는 아래와 같이 나타낼 수 있다.

```
sp20157522025@ubuntu:~/testfolder/testfolder2$ ls -l
total 4
drwxrwxr-x 2 sp20157522025 sp20157522025 4096 Mar 15 23:09 testfolder3
```

```
sp20157522025@ubuntu:~/testfolder/testfolder2$ mkdir newfolder!
sp20157522025@ubuntu:~/testfolder/testfolder2$ ls -l
total 8
drwxrwxr-x 2 sp20157522025 sp20157522025 4096 Mar 16 00:16 newfolder!
drwxrwxr-x 2 sp20157522025 sp20157522025 4096 Mar 15 23:09 testfolder3
```



다음과 같이 directory가 생성된 것을 확인할 수 있다.

(8) rmdir

rmdir은 비어있는 directory를 지우는 명령이다. 설명 그대로 directory가 비어있지 않으면 해당 directory를 지울 수 없다. 결과는 아래와 같이 나타난다.

```
sp20157522025@ubuntu:~/testfolder/testfolder2/newfolder$ ls
newfolder2
```

다음과 같이 newfolder에는 newfolder2라는 폴더가 있다. 작업환경 testfolder2에서 rmdir명령을 쓰면 다음 결과가 나온다.

```
sp20157522025@ubuntu:~/testfolder/testfolder2$ rmdir newfolder
rmdir: failed to remove 'newfolder': Directory not empty
```

Newfolder가 비어있지 않기 때문에 삭제가 불가능하다고 나온다. 해당 directory를 삭제하기 위해서는 다음과 같은 작업을 해주어야 한다.

```
sp20157522025@ubuntu:~/testfolder/testfolder2$ cd newfolder
sp20157522025@ubuntu:~/testfolder/testfolder2/newfolder$ ls
newfolder2
sp20157522025@ubuntu:~/testfolder/testfolder2/newfolder$ rmdir newfolder2
sp20157522025@ubuntu:~/testfolder/testfolder2/newfolder$ ls
sp20157522025@ubuntu:~/testfolder/testfolder2/newfolder$ cd ..
sp20157522025@ubuntu:~/testfolder/testfolder2$ ls
newfolder  testfolder3
sp20157522025@ubuntu:~/testfolder/testfolder2$ rmdir newfolder
sp20157522025@ubuntu:~/testfolder/testfolder2$ ls
testfolder3
```

Newfolder2를 먼저 지워준 후 newfolder를 empty상태로 만들어주고 rmdir명령을 사용하여 정상적으로 명령이 실행 되는 모습을 확인할 수 있다.

(9) rm

rm 명령은 이전에 설명한 rmdir와 비슷한 명령으로 추가적으로 directory만 취급하는 rmdir명령과는 달리 file까지 삭제할 수 있는 명령이다. 단 -r 옵션이 있는 경우 directory를 하위 파일들과 함께 삭제 가능하다.

```
sp20157522025@ubuntu:~/testfolder$ ls
FileA  FileB  testfolder2
sp20157522025@ubuntu:~/testfolder$ rmdir FileA
rmdir: failed to remove 'FileA': Not a directory
sp20157522025@ubuntu:~/testfolder$ rm FileA
sp20157522025@ubuntu:~/testfolder$ ls
FileB  testfolder2
sp20157522025@ubuntu:~/testfolder$
```

다음과 같이 rmdir명령으로는 FileA를 지울 수 없었지만 rm명령으로는 삭제할 수 있는 것을 확인할 수 있다. 또한 rm의 명령은 -r과 -i라는 옵션 명령이 있는데 -r명령은 비어있지 않은 directory도 재귀적으로 안에 내용을 전부 지워주면서 해당 directory를 지워주는 명령이다. -i 명령은 directory에 포함되어있는 파일을 순차적으로 삭제할 것인지 아닌지 사용자에게 물어보며 파일을 쉽게 정리할 수 있다. Directory는 해당 명령에서 제외된다. 각각의 명령은 아래와 같은 결과를 나타낸다.

```
sp20157522025@ubuntu:~/testfolder/testfolder2$ cd tsetls
sp20157522025@ubuntu:~/testfolder/testfolder2/tsetls$ ls
new
sp20157522025@ubuntu:~/testfolder/testfolder2/tsetls$ cd ..
sp20157522025@ubuntu:~/testfolder/testfolder2$ rm -r tsetls
sp20157522025@ubuntu:~/testfolder/testfolder2$ ls
testfolder3
```

-r명령으로 분명 tsetls폴더 안에 new라는 폴더가 있지만 -r을 통해 바로 tsetls폴더를 지우는 것을 확인할 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ ls
FileA  FileB  testfolder2
sp20157522025@ubuntu:~/testfolder$ rm -i *
rm: remove regular empty file 'FileA'? y
rm: remove regular file 'FileB'? y
rm: cannot remove 'testfolder2': Is a directory
sp20157522025@ubuntu:~/testfolder$ ls
testfolder2
```

-i명령으로 위에서 설명한 것과 마찬가지로 y, n을 통해 해당 파일을 지울 것인지 아닌지를 결정한다.

(10) Cp

Cp는 copy명령으로 File이나 directory를 복사하는 명령이다. 해당 명령의 결과는 아래와 같이 나타난다.

```
sp20157522025@ubuntu:~/testfolder$ cp FileC FileCcopy
sp20157522025@ubuntu:~/testfolder$ ls
FileC  FileCcopy  FileD  testfolder2
sp20157522025@ubuntu:~/testfolder$ cat FileC FileCcopy
This is File C!
This is File C!
```

FileC를 FileCcopy라는 이름으로 하나 더 생성하는 명령이다. 내용을 확인해보면 두 파일이 같다는 것을 확인할 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ cp html/* .
sp20157522025@ubuntu:~/testfolder$ ls
file1.txt  file2.txt  file3.txt  html
```

다음은 html의 실행파일을 .(current directory)에 복사하는 것이다. 명령이 실행되면서 각 실행파일들이 복사된 것을 확인할 수 있다.

(11) Mv

Mv는 move로 파일을 이동시키는데 사용하는 명령어다. 해당 명령을 실행하면 아래와 같은 실행 결과를 나타낼 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ ls
file1.txt file2.txt file3.txt html move
sp20157522025@ubuntu:~/testfolder$ mv html move
sp20157522025@ubuntu:~/testfolder$ ls
file1.txt file2.txt file3.txt move
sp20157522025@ubuntu:~/testfolder$ cd move
sp20157522025@ubuntu:~/testfolder/move$ ls
html
sp20157522025@ubuntu:~/testfolder/move$
```

또한 mv는 폴더의 이름을 rename할 수 있는데 아래와 같은 예시가 같은 내용이다.

```
sp20157522025@ubuntu:~/testfolder$ ls
file1.txt file2.txt file3.txt move
sp20157522025@ubuntu:~/testfolder$ mv move move2
sp20157522025@ubuntu:~/testfolder$ ls
file1.txt file2.txt file3.txt move2
sp20157522025@ubuntu:~/testfolder$ cd move2
sp20157522025@ubuntu:~/testfolder/move2$ ls
html
```

아래와 같이 경로를 설정하여 파일을 보내는 것도 가능하다.

```
sp20157522025@ubuntu:~$ ls
Desktop Downloads file1.txt Pictures Templates Videos
Documents examples.desktop Music Public testfolder

sp20157522025@ubuntu:~$ mv file1.txt ./testfolder/move2
sp20157522025@ubuntu:~$ cd testfolder
sp20157522025@ubuntu:~/testfolder$ cd move2
sp20157522025@ubuntu:~/testfolder/move2$ ls
file1.txt html
```

파일을 복사하는 것이 아니라 이동 시키는 것에 주의하자

(12) Ln

ln 명령은 file들간의 링크를 생성시켜주는 명령이다. Cp 명령과의 차이점은 아래와 같이 나타낼 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ ln FileA FileB
sp20157522025@ubuntu:~/testfolder$ ls
FileA  FileB
sp20157522025@ubuntu:~/testfolder$ cat FileA
The is File A
sp20157522025@ubuntu:~/testfolder$ cat FileB
The is File A
```

```
sp20157522025@ubuntu:~/testfolder$ cp FileB FileC
sp20157522025@ubuntu:~/testfolder$ cat FileC
The is File A
```

위와 같이 FileA를 통하여 FileB link명령을 통해 FileC는 copy명령을 통하여 생성했다. 우선 수정전의 내용은 모두 같게 복사가 된 것을 확인할 수 있으며, link된 것과 copy된 것의 차이는 다음과 같은 명령을 통해 확인할 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ vi FileB
sp20157522025@ubuntu:~/testfolder$ cat FileB
This is vi mode!!
sp20157522025@ubuntu:~/testfolder$ cat FileA
This is vi mode!!
sp20157522025@ubuntu:~/testfolder$ cat FileC
The is File A
```

Vi 명령을 통하여 FileB의 내용을 다음과 같이 바꾸고 FileA의 내용을 확인하였을 때 link가 되어 있기 때문에 FileA의 내용도 함께 수정되어 있는 것을 확인할 수 있다. 하지만 FileC는 link된 것이 아니라 copy된 것으로 수정되지 않는 모습을 확인할 수 있다.

또한 ln 명령에 -s 옵션을 포함하여 symbolic link를 만들 수 있다. Symbolic link란 윈도우의 바로가기 기능과 같은 것으로 파일의 위치와 정보만을 기록한 파일이다. 기존에 만든 hard링크와의 차이는 아래와 같다.

```
sp20157522025@ubuntu:~/testfolder$ ln -s FileC FileD
sp20157522025@ubuntu:~/testfolder$ ls
FileA  FileB  FileC  FileD
sp20157522025@ubuntu:~/testfolder$ rm FileA
sp20157522025@ubuntu:~/testfolder$ rm FileC
sp20157522025@ubuntu:~/testfolder$ ls
FileB  FileD
sp20157522025@ubuntu:~/testfolder$ cat FileB
This is vi mode!!
sp20157522025@ubuntu:~/testfolder$ cat FileD
cat: FileD: No such file or directory
sp20157522025@ubuntu:~/testfolder$
```

결과 화면과 같이 FileA와 FileB는 링크되어 있지만 FileA가 삭제된다고 해도 FileB가 없어지지 않는 반면 symbolic link인 C와 D는 C가 없어지자 D를 열수 없게 된다.

(13) Touch

내용이 없는 빈 파일을 만들거나 기존 파일의 시간을 변경하는 명령어로 해당 명령을 사용하면 아래와 같은 결과를 나타낼 수 있다.

```
sp20157522025@ubuntu:~/testfolder$ ls
FileA
sp20157522025@ubuntu:~/testfolder$ touch emptyFile
sp20157522025@ubuntu:~/testfolder$ touch emptyFile2
sp20157522025@ubuntu:~/testfolder$ ls
emptyFile emptyFile2 FileA
```

위 사진은 빈 파일을 생성하는 과정을 나타낸 것이다.

```
sp20157522025@ubuntu:~/testfolder$ ls -l
total 4
-rw-rw-r-- 1 sp20157522025 sp20157522025 0 Mar 16 08:12 emptyFile
-rw-rw-r-- 1 sp20157522025 sp20157522025 0 Mar 16 08:12 emptyFile2
-rw-rw-r-- 1 sp20157522025 sp20157522025 6 Mar 16 08:11 FileA
sp20157522025@ubuntu:~/testfolder$ touch emptyFile2
sp20157522025@ubuntu:~/testfolder$ ls -l
total 4
-rw-rw-r-- 1 sp20157522025 sp20157522025 0 Mar 16 08:12 emptyFile
-rw-rw-r-- 1 sp20157522025 sp20157522025 0 Mar 16 08:14 emptyFile2
-rw-rw-r-- 1 sp20157522025 sp20157522025 6 Mar 16 08:11 FileA
```

위 사진은 파일의 생성 시간을 변경한 이미지다.

(14) Ps

프로세스의 현재 상태를 출력해주는 명령이다. 실습시간에 배운 옵션으로는 -e, -f가 있으며, -e의 뜻은 모든 프로세스를 선택하게 되며, -f는 완벽한 형식에 맞춰 listing이 될 수 있도록 도와준다. 해당 명령의 결과는 다음과 같다.

```
sp20157522025@ubuntu:~$ ps
  PID TTY          TIME CMD
 2930 pts/0        00:00:00 bash
 3141 pts/0        00:00:00 ps
sp20157522025@ubuntu:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1         0  0  07:22 ?           00:00:02 /sbin/init auto noprompt
root           2         0  0  07:22 ?           00:00:00 [kthreadd]
root           4         2  0  07:22 ?           00:00:00 [kworker/0:0H]
root           6         2  0  07:22 ?           00:00:00 [mm_percpu_wq]
root           7         2  0  07:22 ?           00:00:01 [ksoftirqd/0]
root           8         2  0  07:22 ?           00:00:00 [rcu_sched]
root           9         2  0  07:22 ?           00:00:00 [rcu_bh]
root          10         2  0  07:22 ?           00:00:00 [migration/0]
root          11         2  0  07:22 ?           00:00:00 [watchdog/0]
root          12         2  0  07:22 ?           00:00:00 [cpuhp/0]
root          13         2  0  07:22 ?           00:00:00 [kdevtmpfs]
root          14         2  0  07:22 ?           00:00:00 [netns]
root          15         2  0  07:22 ?           00:00:00 [rcu_tasks_kthre]
root          16         2  0  07:22 ?           00:00:00 [kauditd]
root          17         2  0  07:22 ?           00:00:00 [khungtaskd]
root          18         2  0  07:22 ?           00:00:00 [oom_reaper]
```

(15) Pstree

pstree명령은 프로세스를 tree관계로 나타내어 보여주는 명령어다. 해당 명령어의 결과는 아래와 같다.

```
sp20157522025@ubuntu:~$ pstree
systemd--NetworkManager--dhclient
                        --dnsmasq
                        --{gdbus}
                        --{gmain}
--VGAAuthService
--accounts-daemon--{gdbus}
                  --{gmain}
--acpid
--agetty
--avahi-daemon--avahi-daemon
--colord--{gdbus}
         --{gmain}
--cron
--cups-browsed--{gdbus}
               --{gmain}
--cupsd--5*[dbus]
--dbus-daemon
--gnome-keyring-d--{gdbus}
                  --{gmain}
                  --{timer}
--lightdm--Xorg--{InputThread}
          --lightdm--upstart--at-spi-bus-laun--dbus-daemon
```

(16) Exit

exit명령은 셸 스크립트 문장 내에서 스크립트를 종료한다. 다음은 exit의 예시다.

```
sp20157522025@ubuntu:~$ sudo apt-get install csh
[sudo] password for sp20157522025:
Sorry, try again.
[sudo] password for sp20157522025:
Sorry, try again.
[sudo] password for sp20157522025:
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 295 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 (143 kB/s)
Selecting previously unselected package csh.
(Reading database ... 177270 files and directories currently installed.)
Preparing to unpack .../csh_20110502-2.1ubuntu1_amd64.deb ...
```

```
sp20157522025@ubuntu:~$ ps
  PID TTY          TIME CMD
 3227 pts/0        00:00:00 bash
 3667 pts/0        00:00:00 ps
sp20157522025@ubuntu:~$ csh
% ps
  PID TTY          TIME CMD
 3227 pts/0        00:00:00 bash
 3668 pts/0        00:00:00 csh
 3669 pts/0        00:00:00 ps
% exit
% exit
sp20157522025@ubuntu:~$ ps
  PID TTY          TIME CMD
 3227 pts/0        00:00:00 bash
 3670 pts/0        00:00:00 ps
sp20157522025@ubuntu:~$
```

다음은 보면 exit을 통하여 스크립트를 종료한 것을 확인할 수 있다.

(17) Kill

kill 명령어는 프로세스에 종료 시그널을 보낸다. 아래와 같이 다른 터미널에서 문자가 반복적으로 출력될 때 다른 터미널을 실행하여 kill 명령을 통해 다른 터미널의 동작을 멈출 수 있다.

[illegible]

또한 kill 명령은 강제종료 옵션이 있는데 해당 명령은 아래와 같은 예제로 볼 수 있다.

```

sp20157522025@ubuntu: ~
sp20157522025@ubuntu:~$ ps
  PID TTY          TIME CMD
 3778 pts/0    00:00:00 bash
 3788 pts/0    00:00:00 ps
sp20157522025@ubuntu:~$ vi hello

[1]+  Stopped                  vi hello
sp20157522025@ubuntu:~$ ps
  PID TTY          TIME CMD
 3778 pts/0    00:00:00 bash
 3789 pts/0    00:00:00 vi
 3790 pts/0    00:00:00 ps
sp20157522025@ubuntu:~$ kill -9 3789

[1]+  Killed                  vi hello
sp20157522025@ubuntu:~$ ps
  PID TTY          TIME CMD
 3778 pts/0    00:00:00 bash
 3793 pts/0    00:00:00 ps

```

위와 같이 vi편집기가 정상적으로 종료되지 않아 kill -9 명령을 통하여 종료했다.

(18) Time

time명령어는 특정 프로그램이나 명령어를 인자로 실행한다. 포맷 형식을 지정하지 않으면 real, user, sys 정보를 출력하게 된다. 아래는 그에 해당하는 예시다.

```
sp20157522025@ubuntu:~$ time ps
  PID TTY          TIME CMD
 2023 pts/4        00:00:00 bash
 2057 pts/4        00:00:00 ps

real    0m0.008s
user    0m0.004s
sys     0m0.004s
```

Real: 실제 cpu 소요 시간

User: user영역에서 소비된 cpu 시간

Sys: 커널에서 소비된 cpu 시간

(19) Passwd

사용자의 비밀번호를 바꿔주는 명령어다. 명령을 실행하여 기존 로그인하는 비밀번호를 변경할 수 있으며, 아래는 그에 해당하는 결과다.

```
sp20157522025@ubuntu:~$ passwd
Changing password for sp20157522025.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```


(20) Uname

Uname명령은 시스템에 대한 정보를 출력하는 역할을 한다. 옵션을 지정하지 않으면 -s 옵션과 같이 kernel이름을 출력한다.

```
sp20157522025@ubuntu:~$ uname
Linux
sp20157522025@ubuntu:~$ uname -r
4.15.0-29-generic
sp20157522025@ubuntu:~$ uname -m
x86_64
sp20157522025@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
```

-r : kernel release정보를 출력한다.

-m : 시스템의 하드웨어 타입을 출력한다.

-a : 모든 정보를 출력하게된다.

(21) Wc

wc명령은 문서의 행과 단어 문자개수를 카운트하고 출력하는 기능을 한다. 옵션으로 -c, -w, -l이 있으며, -c는 바이트 수만큼 출력하고, -w는 단어 수만큼 출력하며, -l은 행 수를 출력한다. 아래는 파일 A에 대한 명령어 결과다.

```
sp20157522025@ubuntu:~/testfolder$ cat A
aaaaaaaaabbbbbbbbbbcccccc

dasdasds
sp20157522025@ubuntu:~/testfolder$ wc A
 7  2 44 A
sp20157522025@ubuntu:~/testfolder$ wc -c A
44 A
sp20157522025@ubuntu:~/testfolder$ wc -w A
 2 A
sp20157522025@ubuntu:~/testfolder$ wc -l A
 7 A
```

(22) More

more 명령은 한 페이지 이상 되는 내용을 한 화면의 페이지 단위로 보여주며 다양한 기능을 제공한다.

[illegible]

(23) Echo

Echo 명령은 지정한 문자열을 출력하는 명령어다. 참고로 문자열과 개행 문자를 덧붙여 출력한다. 옵션으로 개행을 붙이지 않을 수도 있다.

```
sp20157522025@ubuntu:~$ echo helloworld
helloworld
sp20157522025@ubuntu:~$ echo $HOME
/home/sp20157522025
sp20157522025@ubuntu:~$ echo ~
/home/sp20157522025
```

(24) Alias

Alias는 기존에 있는 명령어를 사용자가 지정하는 문자에 동작할 수 있도록 허용해주는 명령을 뜻한다. 아래는 예시다.

```
sp20157522025@ubuntu:~$ myls
No command 'mysl' found, did you mean:
  Command 'tyls' from package 'terminology' (universe)
  Command 'mmls' from package 'sleuthkit' (universe)
mysl: command not found
sp20157522025@ubuntu:~$ alias mysl='ls -al'
sp20157522025@ubuntu:~$ mysl
total 132
drwxr-xr-x 16 sp20157522025 sp20157522025 4096 Mar 16 17:41 .
drwxr-xr-x  3 root          root          4096 Mar 15 19:55 ..
-rw-r--r--  1 sp20157522025 sp20157522025 4894 Mar 16 18:13 .bash_history
-rw-r--r--  1 sp20157522025 sp20157522025  220 Mar 15 19:55 .bash_logout
-rw-r--r--  1 sp20157522025 sp20157522025 3771 Mar 15 19:55 .bashrc
drwxr-xr-x 12 sp20157522025 sp20157522025 4096 Mar 16 00:13 .cache
drwxr-xr-x 15 sp20157522025 sp20157522025 4096 Mar 15 22:55 .config
drwxr-xr-x  2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Desktop
-rw-r--r--  1 sp20157522025 sp20157522025   25 Mar 15 20:33 .dmrc
drwxr-xr-x  2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Documents
drwxr-xr-x  2 sp20157522025 sp20157522025 4096 Mar 15 20:33 Downloads
-rw-r--r--  1 sp20157522025 sp20157522025 8980 Mar 15 19:55 examples.desktop
drwxr-xr-x  2 sp20157522025 sp20157522025 4096 Mar 15 22:08 .gconf
```

Mysl로 지정하여 ls -a를 실행하는 모습이다.

```
sp20157522025@ubuntu:~$ 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'
```

Alias를 통하여 alias로 지정된 명령을 확인할 수 있다.

```
sp20157522025@ubuntu:~$ unalias mysl
sp20157522025@ubuntu:~$ 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'
```

또한 unalias로 지정한 명령을 해제할 수 있다.

(25) Find

Find는 주어진 조건에 따라 디렉토리를 검색해서 원하는 파일을 찾는 명령어다. 현재는 간단한 예시지만 실제로 find는 강력한 기능을 제공한다.

```
sp20157522025@ubuntu:~$ cd testfolder
sp20157522025@ubuntu:~/testfolder$ ls
A.txt B.txt C.txt
sp20157522025@ubuntu:~/testfolder$ cd ..
sp20157522025@ubuntu:~$ find -name '*.txt'
./testfolder/B.txt
./testfolder/C.txt
./testfolder/A.txt
```

(26) grep

grep명령어는 지정한 특정 문자열을 검색하여 동일한 문자열이 있는 줄의 패턴을 찾아 화면에 출력해주는 명령이다. 아래는 명령어의 예시다.

```
sp20157522025@ubuntu:~/testfolder$ cat A.txt
aaaaaaaaabbbbbbbbbbcccccc

dasdasds
sp20157522025@ubuntu:~/testfolder$ grep a A.txt
aaaaaaaaabbbbbbbbbbcccccc
dasdasds
```

6. Reference

실습강의자료와 네이버 지식백과를 통하여 명령어의 대한 옵션과 기능을 이해하는데 참고하였습니다. 특히 chmod는 아래 링크를 참고하여 이해하였습니다.

Chmod 참고 : <https://hjleesm.blog.me/221350302093>

1차 과제 – Linux 기초

1-3차: Usage of Linux Commands

시스템 프로그래밍 실습

제출일: 3월 29일 금요일

분 반: 화요일

담당 교수: 신영주

학 번: 2015722025

학 과: 컴퓨터정보공학부

이 름: 정용훈

7. Introduction

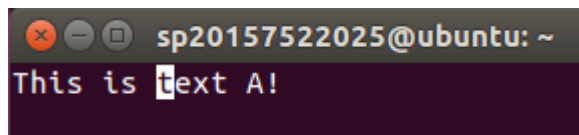
2주차에 이어 기본 명령어인 vi의 삽입, 삭제, 데이터 저장 방법을 익히고, make와 gdb명령의 사용방법을 익힌다.

8. Result

(1) Vi

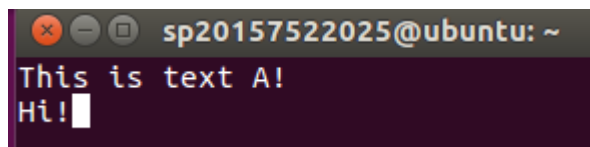
삽입

Vi모드에서 삽입은 여러 가지가 있지만 실습으로는 o를 사용한 삽입을 실행해보았다. o는 현재 커서가 위치한 아래 행으로 이동하여 문장을 삽입하게 된다. 실행 결과는 다음과 같다.



```
sp20157522025@ubuntu: ~  
This is text A!
```

명령 모드에서 o실행

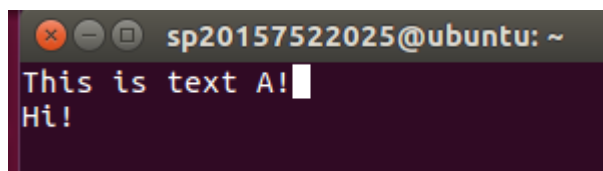


```
sp20157522025@ubuntu: ~  
This is text A!  
Hi!
```

커서가 다음 행으로 넘어가며, 문장 삽입이 가능해진다.

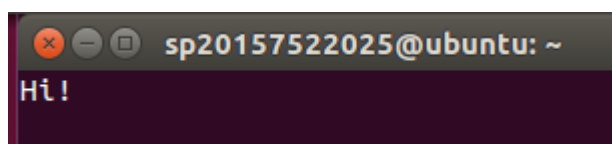
삭제

다음으로 실행할 명령어는 dd로 커서가 있는 한 행을 삭제하는 명령이다. 다음 명령을 실행하면 다음과 같이 결과를 볼 수 있다.



```
sp20157522025@ubuntu: ~  
This is text A!  
Hi!
```

명령 모드에서 dd실행




```
sp20157522025@ubuntu: ~  
Hi!
```

데이터 저장

다음은 vi모드 실행 후 내용을 생성하거나 수정한 후 해당 데이터를 저장 할 때 사용하는 명령이다. 알아볼 명령어는 wq로 파일을 저장 후 종료하는 명령이다.

```
sp20157522025@ubuntu:~/test$ cat A.txt
Hi i'm A!
```

다음은 원래 A파일의 내용이다.



The screenshot shows a terminal window with a dark background. The title bar at the top contains three window control icons (close, minimize, maximize) and the text "sp20157522025@ubuntu: ~/test". The main content of the terminal is the text "This is File A!" followed by 15 tilde characters (~) arranged in a vertical column. At the bottom left of the terminal, the text ":wq" is visible.

Vi 모드 진입 후 각종 명령을 통하여 문장을 수정한 후 명령 모드에서 "shift + ." 입력 후 wq를 실행한다.

```
sp20157522025@ubuntu:~/test$ vi A.txt
sp20157522025@ubuntu:~/test$ cat A.txt
This is File A!
```

다음과 같이 파일 A의 내용이 바뀌어 저장된 것을 확인할 수 있다.

(2) Make

Make 명령은 컴파일 과정을 자동화하기 위해 사용하며, 특히 gcc컴파일러의 다양한 옵션들을 컴파일 할 때마다 많은 시간이 소요되기 때문에 편의성을 위하여 필요하다. make명령을 사용하기 위해서는 Makefile이 있어야 하는데, makefile은 컴파일 할 소스 파일과 컴파일 옵션에 관련하여 정의 해놓은 스크립트 파일이다. 아래 과정은 makefile을 생성하는 것과 make명령을 실행하였을 때 결과다.

```
sp20157522025@ubuntu: ~/test
#include<stdio.h>

int main()
{
printf("Test\n");
}
~
~
~
```

다음은 컴파일 할 코드다.

```
sp20157522025@ubuntu: ~/test
all:
    gcc -o run A.c
~
```

make명령을 쓰기 위한 Makefile의 내용이다. A.c파일을 컴파일 하여 run이라는 실행 파일을 만들고 해당 실행파일을 실행하게 되면 컴파일 된 결과를 터미널에서 아래와 같이 바로 볼 수 있다.

```
sp20157522025@ubuntu:~/test$ make
gcc -o run A.c
sp20157522025@ubuntu:~/test$ ls
A.c Makefile run
sp20157522025@ubuntu:~/test$ ./run
Test
```


다음은 Makefile을 통해 여러 개의 c파일을 한 개의 실행파일이 아닌 여러개의 실행 파일로 만드는 과정이다. 해당 과정은 아래와 같다.

```
sp20157522025@ubuntu:~/test2$ cat Makefile
all:
    gcc A.c -o runA
    gcc B.c -o runB
A: A.c
    gcc A.c -o runA
B: B.c
    gcc B.c -o runB
```

Make file을 다음과 같이 작성하여 make명령에 인자가 없을 경우 c파일을 컴파일 하여 runA, runB라는 실행파일을 각각 만들게 된다.

```
sp20157522025@ubuntu:~/test2$ make
gcc A.c -o runA
gcc B.c -o runB
sp20157522025@ubuntu:~/test2$ ls
A.c B.c Makefile runA runB
```

실행파일이 생성된 모습이며, 각각 실행파일을 실행하면 다음과 같다.

```
sp20157522025@ubuntu:~/test2$ ./runA
This is file A!
sp20157522025@ubuntu:~/test2$ ./runB
This is File B!
```

마지막으로 터미널에서의 명령을 매개변수로 값을 전달하는 것도 가능하다. 그에 해당하는 예시는 다음과 같다.

```
sp20157522025@ubuntu:~/test$ cat A.c
#include<stdio.h>

int main(int argc, char*argv[])
{
    printf("The Number of Inputted Variable is %d\n",argc);
    printf("and they are ");

    for(int i=0; i<argc;i++)
    {
        printf("%s", argv[i]);
    }

    printf("\n");

    return 0;
}
sp20157522025@ubuntu:~/test$ ./run Hi This is test!
The Number of Inputted Variable is 5
and they are ./runHiThisistest!
```

입력인자들의 개수를 int로 받으며, char*로 인자들을 배열 형으로 저장한다.

(3) Gdb

다음은 gdb명령이다. gdb명령은 간단하게 말하면 디버깅을 하는 명령이라고 생각하면 쉽지만, 조금 더 강력한 기능들이 많다. gdb명령을 실행하기 위해선 makefile도 기존과는 다르게 작성해야 하는데 다음과 같은 과정을 거친다.

```
sp20157522025@ubuntu:~/test$ cat Makefile
all:
    gcc -o run -g A.c
```

다음과 같이 실행파일 앞에는 -o, 소스 파일 앞에는 -g 가 반드시 있어야 gdb에서의 기능을 사용할 수 있다.

```
sp20157522025@ubuntu:~/test$ gdb run
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from run...done.
(gdb) █
```

다음은 gdb모드를 실행한 결과 창이다. Gdb의 명령은 실습 자료에 많이 나와있으며, 실제로 실습한 명령은 list, r, s, n이며, 과정은 다음과 같다.

```
(gdb) list
1      #include<stdio.h>
2
3      int main()
4      {
5          int i;
6          double j;
7
8          for(i=0;i<5;i++)
9          {
10             j=i/2+i;
(gdb) list
11         printf("j is %f \n",j);
12     }
13 }
(gdb) █
```

list명령을 통하여 코드의 내용을 10줄씩 확인할 수 있다.

```
(gdb) b 9
Breakpoint 1 at 0x400537: file A.c, line 9.
```

b 9 명령을 통해 9번째 줄에서 breaking point가 걸린 모습이다.

```
(gdb) r
Starting program: /home/sp20157522025/test/run

Breakpoint 1, main () at A.c:10
10      j=i/2+i;
(gdb) n
11      printf("j is %f \n",j);
(gdb) n
j is 0.000000
8      for(i=0;i<5;i++)
(gdb) n

Breakpoint 1, main () at A.c:10
10      j=i/2+i;
(gdb) n
11      printf("j is %f \n",j);
(gdb) q
A debugging session is active.

        Inferior 1 [process 3691] will be killed.

Quit anyway? (y or n) █
```

r명령으로 프로그램을 시작하고, n명령을 통하여 한 줄씩 코드를 실행시키는 모습이다. 실행하는데 있어 s명령도 있지만 s명령은 함수를 무시 하지 않고 해당함수로 넘어가기 때문에 n으로 한 줄씩 실행 시킨 모습이다. 마지막으로 q명령을 통해 gdb를 종료한 모습이다.

다음은 watchpoint라 정의되는 특정 식의 값이 변경되거나 읽혀질 때 프로그램의 수행이 stop되는 특별한 breakpoint다. 아래는 실행의 예시다.

```
(gdb) watch j
Hardware watchpoint 2: j
(gdb) c
Continuing.
j is 0.000000

Breakpoint 1, main () at A.c:10
10      j=i/2+i;
(gdb) c
Continuing.

Hardware watchpoint 2: j

Old value = 0
New value = 1
main () at A.c:11
11      printf("j is %f \n",j);
(gdb) c
Continuing.
j is 1.000000

Breakpoint 1, main () at A.c:10
10      j=i/2+i;
```

j변수를 watch명령으로 설정 후 c명령을 통해 실행하여 j의 값이 바뀌면 stop이 되는 것을 확인할 수 있다.

마지막으로 다음과 같은 명령으로 변수들의 정보도 볼 수 있다.

```
(gdb) info locals
i = 2
j = 1
```

9. Reference

3번째 과제는 실습자료와 강의를 통해 충분히 해결할 수 있었다.