- **명령어 cat**

기본적인 쓰임은 파일을 불러와서 출력하는 것.

파일 출력할 때 줄 번호 생성

```
s21800201@walab-HGU:~/ossl_lab5$ cat -b test.c
     1  #include <stdio.h>
     2  int main() {
     3          printf("Shalom !!!!₩n");
     4          printf("Name: 김현욱 ₩n");
     5          return 0;
     6  }
s21800201@walab-HGU:~/ossl_lab5$ cat -n test.c
     1  #include <stdio.h>
     2  int main() {
     3          printf("Shalom !!!!₩n");
     4          printf("Name: 김현욱 ₩n");
     5          return 0;
     6  }
```

Tab을 ^I로 표현해서 보여줌

```
s21800201@walab-HGU:~/ossl_lab5$ cat -T test.c
#include <stdio.h>
int main() {
^Iprintf("Shalom !!!!₩n");
^Iprintf("Name: 김현욱 ₩n");
^Ireturn 0;
}
```

파일 두 개 연속으로 출력하기

```
s21800201@walab-HGU:~/ossl_lab5$ cat test.c test2.c
#include <stdio.h>
int main() {
        printf("Shalom !!!!₩n");
        printf("Name: 김현욱 ₩n");
        return 0;
}
#include <stdio.h>

int main() {
        printf("My Name is 현욱 ₩n");
        return 0;
}
```

- **명령어 sort**

줄 번호를 임의로 섞어서 출력

```
s21800201@walab-HGU:~/ossl_lab5$ sort test.c
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        return 0;
#include <stdio.h>
int main() {
}
```

```
s21800201@walab-HGU:~/ossl_lab5$ sort -d test.c
}
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        return 0;
#include <stdio.h>
int main() {
s21800201@walab-HGU:~/ossl_lab5$ sort -R test.c
        printf("Shalom !!!!₩n");
int main() {
        printf("Name: 김현욱 ₩n");
        return 0;
}
#include <stdio.h>
```

- **명령어 uniq**

중복된 내용의 행이 연속으로 있으면 하나만 남기고 삭제한다.

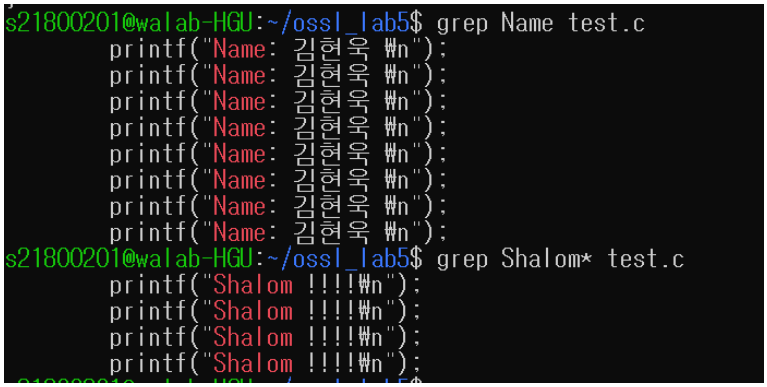
전체적으로 분산된 중복은 찾아내지 못하므로 정렬하여 순차적으로 만든 뒤 적용

```
s21800201@walab-HGU:~/ossl_lab5$ s21800201@walab-HGU:~/ossl_lab5$ cat test.c
#include <stdio.h>
int main() {
        printf("Shalom !!!!₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        return 0;
}
s21800201@walab-HGU:~/ossl_lab5$ sort test.c
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        return 0;
#include <stdio.h>
int main() {
}
s21800201@walab-HGU:~/ossl_lab5$ uniq test.c
#include <stdio.h>
int main() {
        printf("Shalom !!!!₩n");
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        return 0;
}
```

- **명령어 grep**

파일이나 표준 입력으로부터 패턴을 찾아주는 역할을 한다.

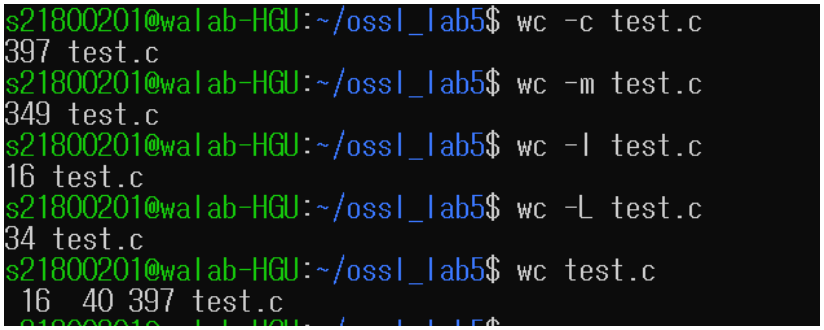
특정 문자열을 찾는 데 가장 유용하게 쓰인다.



- **명령어 wc**

주어진 파일의 바이트, 문자, 단어, 줄 수를 출력해주는 명령어

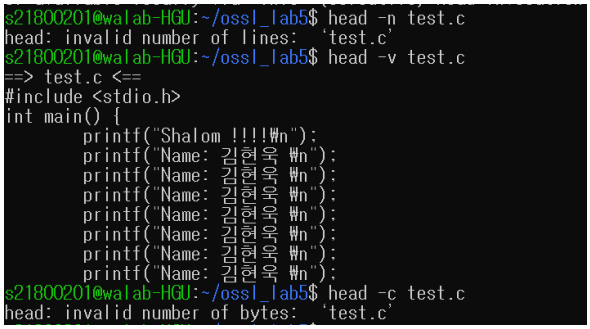
Word count의 약자



-c는 byte -m은 문자 개수 -l은 줄 개수,

- **명령어 head**

텍스트로 된 파일의 앞부분을 지정한 만큼 출력하는 명령어



- 명령어 tail

어떠한 오류나 파일의 로그를 보고싶을 경우 사용함

파일의 마지막 부분을 출력한다.

```
s21800201@walab-HGU:~/ossl_lab5$ tail test.c
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        return 0;
}
```

```
s21800201@walab-HGU:~/ossl_lab5$ tail -v test.c
==> test.c <==
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Name: 김현욱 ₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        printf("Shalom !!!!₩n");
        return 0;
}
```

- 명령어 tee

화면 출력과 텍스트 저장을 동시에 하는 명령어

```
s21800201@walab-HGU:~/ossl_lab5$ tee test.c
Handong Global University
Handong Global University
Hello!!
Hello!!
s21800201@walab-HGU:~/ossl_lab5$ cat test.c
Handong Global University
Hello!!
```

- 명령어 echo

화면에 텍스트를 출력하는 명령어

간단히 파일을 만들 때 사용할 수도 있음

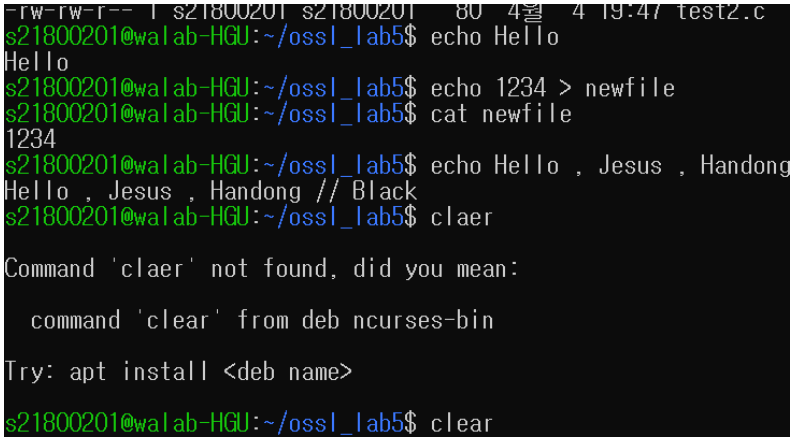
```
s21800201@walab-HGU:~/ossl_lab5$ echo > newfile
s21800201@walab-HGU:~/ossl_lab5$ ls
Makefile  Makefile_macro  guest.c  guest.h  main.c  menu.c  menu.h  newfile
```

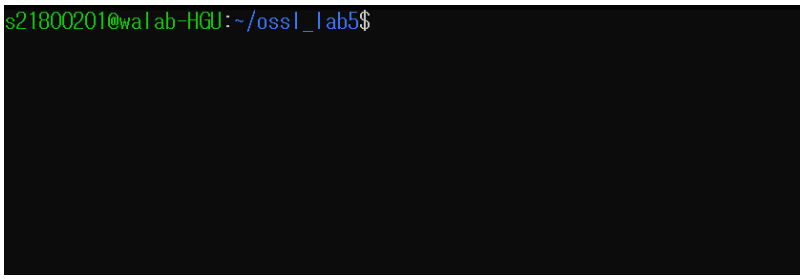
```
s21800201@walab-HGU:~/ossl_lab5$ echo 1234 > newfile
s21800201@walab-HGU:~/ossl_lab5$ cat newfile
1234
```

```
s21800201@walab-HGU:~/ossl_lab5$ echo Hello , Jesus , Handong // Black
Hello , Jesus , Handong // Black
```

- 명령어 clear

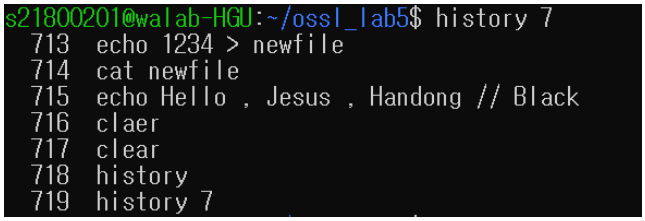
터미널의 내용을 모두 지우는 명령어





- 명령어 history

이전에 실행했던 명령어는 화살표 위, 아래로 쉽게 복구할 수 있는 데 이 이유는 history 에 저장되어 있기 때문이다.



- 명령어 id

리눅스 명령어 id는 현재 사용자의 실제 id와 유효 사용자 id, 그룹 id를 출력해준다.

- 명령어 chmod

Change mode의 약자로 대상 파일과 디렉토리의 사용권한을 변경할 때 사용한다.

- 명령어 umask

umask명령어는 기본 허가권이 어떻게 세팅되어있는지 확인할 수 있는 명령어

- 명령어 su

현재 계정을 로그아웃 하지 않고 다른 계정으로 전환하는 명령어

- 명령어 sudo

sudo명령어는 유닉스, 리눅스 계열에서 다른 사용자의 보안권한과 관련된 프로그램을 구동할 수 있게 해주는 명령어

- 명령어 chown

chown명령어는 파일의 소유자를 변경시켜 주는 명령어

- 명령어 chgrp

파일이나 디렉토리의 소유그룹을 변경시키는 명령어

- 명령어 ps

현재 실행 중인 프로세스 목록과 상태를 보여주는 명령어

- 명령어 top

실시간으로 CPU 사용률을 체크해주는 명령어

- 명령어 jobs

백그라운드로 실행되는 작업목록을 보여주는 리눅스 명령어

- 명령어 bg

Background의 약어로 현재 실행중인 프로세스를 백그라운드 작업으로 전환하는 명령어

- 명령어 fg

현재 백그라운드로 실행중인 명령어를 포그라운드 작업으로 전환할 때 사용하는 명령어

- 명령어 kill

프로세스를 강제로 종료시키는 명령어가 아닌 프로세스에 시그널을 보내는 명령어

시스템에 문제가 생겨 해당 프로세스를 터미널에서 종료시킬 경우 유용함

- 명령어 killall

프로세스 번호가 아닌 이름으로 프로세스를 종료시키는 명령어

- 명령어 shutdown

시스템 종료, 시스템 리부팅, 즉시 리부팅 등 시스템 종료와 관련된 명령어

1. 홈디렉터리에 2021OSS디렉터리 생성 및 이동

필요한 명령어: mkdir, cd

2021OSS 디렉토리를 mkdir명령어와 함께 mkdir 2021OSS 로 생성한다.

해당 디렉토리로의 이동은 cd명령어와 함께 cd 2021OSS 하면 디렉토리가 이동된다.

2. 2. 2021OSS디렉터리내에 lab5 디렉터리 생성 이동

필요한 명령어: mkdir, cd

Lab5디렉터리를 mkdir 명령어와 함께 mkdir lab5 로 생성한다.

해당 디렉토리로의 이동은 cd명령어와 함께 cd lab5 하면 디렉토리가 이동한다.

3. 홈디렉터리에 lab5 디렉터리를 바로 갈 수 있는 symbolic link 생성(이름 : ossl_lab5)

필요한 명령어: ln -s

Symbolic link의 생성은 ln -s 원본파일 링크이름의 방식으로 하며 ln -s 2021OSS/lab5 ossl_lab5 로 링크를 생성하면 된다.

4. 홈디렉터리에서 ossl_lab5 이용해서 lab5 디렉터리로 이동

필요한 명령어: cd

디렉토리의 이동은 cd ossl_lab5를 통해서 해당 symbolic link가 가리키고 있는 원본 파일의 경로로 이동하게 된다.

5. vim test.c 실행 -> 입력모드 변경

필요한 명렁어: vim

vim으로 실행하기 위해 필요한 키보드 키: i

Vim 을 통해 test.c 파일을 실행하고 i를 통해 입력모드를 변경하게 된다.

6. 아래 내용 입력

vim으로 실행하기 위해 필요한 키보드 키: i, hjkl(방향키)

입력하기 위해서는 i를 통해 입력을 해야한다.

7. 저장 후 종료

vim으로 실행하기 위해 필요한 키보드 키: esc, ZZ

입력모드에서 Normal 모드로 돌아오기 위해 esc를 입력하고 ZZ를 통해 저장하고 종료한다.

8. Vim test.c 실행

필요한 명령어: vim

Vim test.c 를 입력해서 해당 파일을 연다.

9. 명령모드 변경 => 소스에 줄 번호 보이기 :set number 또는 :set nu 입력

vim으로 실행하기 위해 필요한 키보드 키: :set number, :set nonumber

Test.c 파일 변경 내용: 줄 번호가 생김

Vim test.c를 통해 파일에 들어가서 Normal 모드에서 :set nu를 통해 줄 번호를 만든다.

:set nonumber를 통해 줄번호를 없앨 수 있다.

10. 2번 라인에 커서 위치하고 윗줄에 공백라인 추가

vim으로 실행하기 위해 필요한 키보드 키: O

Test.c 파일 변경 내용: 2번 라인에 공백 라인이 생김

윗줄에 공백라인 추가는 대문자 o, O를 통해 할 수 있다.

11. 3번라인 { 뒤에 enter 입력

vim으로 실행하기 위해 필요한 키보드 키: h,j,k,l,i

Test.c 파일 변경 내용: 4번 라인에 공백 라인이 생김

Normal모드에서 이동 명령어를 이용해서 { 뒤로 이동하고 i를 통해 입력모드로 바꾼 후 enter를 입력한다.

12. 4번 라인에 입력

vim으로 실행하기 위해 필요한 키보드 키: h,j,k,l i

Test.c 파일 변경 내용: 4번 라인에 Hello World 출력문 생성

Normal모드에서 이동 명령어를 이용해 4번 라인으로 이동하고 i를 통해 입력모드로 바꾼 후 tap 키와 해당 코딩을 진행한다.

13. 저장 후 종료

vim으로 실행하기 위해 필요한 키보드 키: esc, ZZ

Esc를 통해 Normal 모드로 이동하고 ZZ를 통해 저장하고 종료한다.

14. 컴파일

필요한 명령어: gcc

15. 현재 디렉터리 조회하여 test파일 생성 확인

필요한 명령어: ls -al

16. 현재 디렉터리에서 test 실행

필요한 명령어: ./test

17. vim test.c

18. printf 라인에 커서 위치

vim으로 실행하기 위해 필요한 키보드 키: h,j,k,l

이동 명령어 h,j,k,l을 통해 라인 이동을 한다.

19. printf 라인 복사

vim으로 실행하기 위해 필요한 키보드 키: yy

20. 두번 붙여넣기

vim으로 실행하기 위해 필요한 키보드 키: p

Test.c 파일 변경 내용: 4,5번 라인에 Hello World 복사됨

21. 한번 실행취소

vim으로 실행하기 위해 필요한 키보드 키: u

Test.c 파일 변경 내용: 기존 4번 라인 제거되고 Hello World는 4번 5번 두개만 존재

22. 두번째 printf(5번행) 문자열에서 한 단어씩 삭제(2번) !!!는 한개씩 삭제

vim으로 실행하기 위해 필요한 키보드 키: dw, x

Test.c 파일 변경 내용: 5번 줄에 있는 Hello와 World가 제거되고 !는 하나씩 모두 제거

단어 제거는 dw를 통해서 단어를 지우고 x를 통해서 한 글자씩 지울 수 있다.

23. 두번째 printf(5번행) 문자열에 자신의 이름 입력후 저장/종료

필요한 명령어: ZZ

Test.c 파일 변경 내용: 5번 줄에 자신의 이름 작성

자신의 이름을 입력하고 ZZ를 통해 저장/종료를 한다.

24. 컴파일(위화살표키로 history검색후 실행)

25. 현재 디렉터리에서 test 실행 및 결과확인

필요한 명령어: ./

26. vim test.c

필요한 명령어: vim

27. Hello를 Shalom(원하는 단어로)치환 후 World단어 삭제 / 저장 / 컴파일 / 실행

필요한 명령어: cw, dw, ZZ

Test.c 파일 변경 내용: 4번 줄 내용이 Shalom World로 바뀜

커서를 H위에 위치시키고 cw를 통해 단어를 변경하고 world에서 dw를 단어를 삭제한다.

28. 자신의 이름라인 복사해서 붙여넣기

필요한 명령어: yy, dd, P

Test.c 파일 변경 내용: 내용이 변하지는 않았지만 복사하고 다시 붙여넣는 과정

이름 라인 복사 명령어는 yy를 통해서 한 줄을 복사하고 dd를 통해서 해당 라인을 제거한다. 그리고 P를 통해서 2번째 라인에 다시 붙여넣을 수 있다.

29. 자신이 좋아하는 명언/성경구절/하고싶은 말 입력

Test.c 파일 변경 내용: 6번 줄에 성경 구절로 채워넣음

vim으로 실행하기 위해 필요한 키보드 키: hjkl(방향키), i,

30. 저장/컴파일/실행

필요한 명령어: gcc, ./