hw

0) Set up bash startup file: copy .bashrc and .bash\_profile into your login directory

$ pwd

/home/sp11/12181234 -- make sure you are in the login directory

$ cp ../../linuxer1/.bashrc **.** (use linuxer2 instead of linuxer1 in 152 server)

$ cp ../../linuxer1/.bash\_profile **.** (use linuxer2 instead of linuxer1 in 152 server)

**And close your putty window and reopen**. Now the prompt should have been changed to

[12181234@linuxer2 ~]$

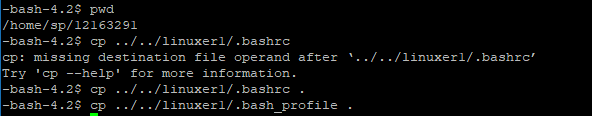
Now you can type program name without "./" prefix.

$ ex1

instead of

$./ex1

The new bash startup files will include the current directory (".") in the PATH environment variable which in turn allows you to type program name withtout "./" prefix to run it. (You have to reopen putty terminal to see the effect!)



…/.bashhrc 와 …/.bash\_profile을 현재 폴더에 복사하였습니다



Shell 프롬프트 창의 내용이 바뀜과 동시에, ./ex1 과달리 이름만 쳐도 컴파일이 실행되는 것을확인할 수 있었습니다.

1) Try following program which doesn't receive command line arguments.

**ex0.c**:

void main(){ // this program doesn't receive command line arguments

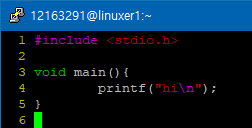
printf("hi\n");

}

$ gcc –o ex0 ex0.c

$ ex0

hi





./ 없이 파일이름만으로 실행이 가능하게 되었습니다.

2) Try following program that receives one command line argument.

**ex1.c**:

void main(int argc, char \* argv[]){ // this program receives command line arguments

printf("hi\n");

printf("%d\n", argc); // number of arguments: 1

printf("%s\n", argv[0]); // the first argument: program name

}

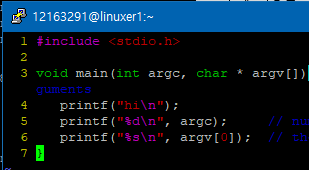
$ gcc –o ex1 ex1.c

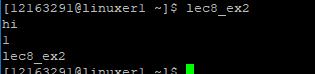
$ ex1

hi

1

ex1





argc 값은 argument의 수로 - 1,

argv[0] 값은 ‘프로그램의 이름’인 것을 확인할 수 있었습니다.

3) Try following program that receives two command line arguments.

**ex2.c**:

void main(int argc, char \* argv[]){ // this program receives command line arguments

printf("hi\n");

printf("%d\n", argc); // number of arguments: 2

printf("%s\n", argv[0]); // the first argument: program name

printf("%s\n", argv[1]); // the second command line argument

}

$ gcc –o ex2 ex2.c

$ ex2

hi

1

ex2

Segmentation fault (core dumped)

=> You have to provide two command line arguments!

$ ex2 hello

hi

2

ex2

hello

$ ex2 hello uzbek tuit

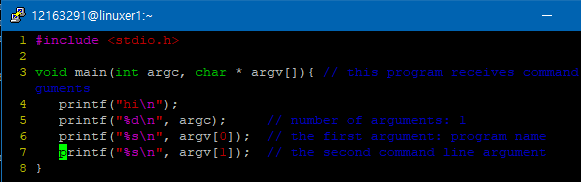
hi

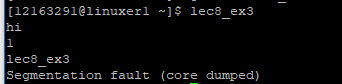
4

ex2

hello

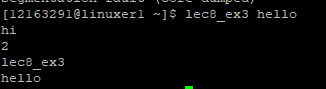
=> If you provide more arguments than what the program can receive, the rest will be ignored.



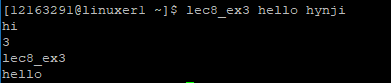


command line argu가 2줄이므로, argc가 2가 되도록, 해주어야합니다.

지금은 1이므로 core dumped가 뜨게 됩니다.



뒤에 hello를 추가하여주니, core dumped되지 않았습니다.



hello 뒤 (hynji) 는 무시됩니다. - 저장된 주소에 있는 내용까지만 출력합니다.

4) A program that receives three command line arguments.

**ex3.c**:

void main(int argc, char \* argv[]){

printf("hi\n");

printf("%d\n", argc);

printf("%s\n", argv[0]); // the first command line argument . the program name

printf("%s\n", argv[1]); // the second command line argument

printf("%s\n", argv[2]); // the third command line argument

}

$ ex3 hello there

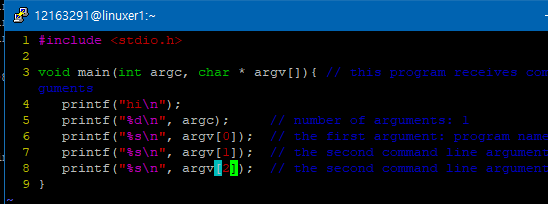
hi

3

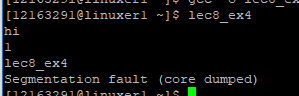
ex3

hello

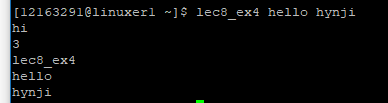
there



3) 번 과제와 거의 동일 합니다.



역시 파일 이름 하나만 입력하니 core dumped 됩니다.



아까는 무시됐던 hynji -3번쨰 값이 잘 뜨는 것을 확인할 수 있었습니다.

5) A program that receives any number of arguments.

**ex4.c**:

void main(int argc, char \*argv[]){

int i;

pirntf("argc is %d\n", argc);

for(i=0;i<argc;i++){

printf("argv[%d] is %s\n", i, argv[i]);

}

}

Run above program with some arguments.

$ ex4 x1 x2 x3 x4

argc is 5

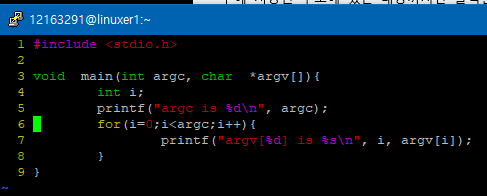
argv[0] is ex4

argv[1] is x1

argv[2] is x2

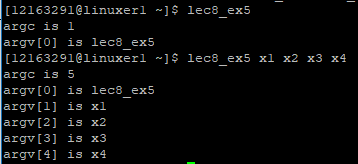
argv[3] is x3

argv[4] is x4



이제는, for문을 통해 커맨드 개수를 자동으로 인지하여,

어떤 커맨드를 입력하던 간에 core dumped가 발생하지 않게 됩니다.



그 결과를 확인할 수 있었습니다.

6) Try following and explain the difference from echo command.

**myecho.c**

void main(int argc, char \*argv[]){

int i;

for(i=1;i<argc;i++){ // skip program name

printf("%s ", argv[i]); // and display all the arguments

}

printf("\n");

}

$ gcc –o myecho myecho.c

$ myecho hello

hello

$ echo hello

hello

$ myecho hello hi bye

hello hi bye

$ echo hello hi bye

hello hi bye

$ echo hi > f1

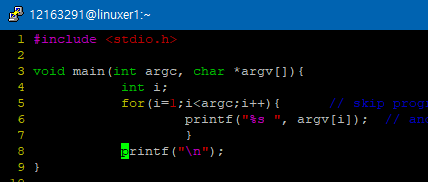
$ cat f1

hi

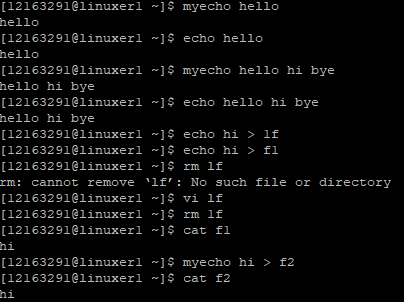
$ myecho hi > f2

$ cat f2

hi



argv[0] 는 프로그램 이름이므로, 제외해줍니다.



myecho 와 echo가 완전히 동일하게 돌아가고 있는 것을 확인할 수 있었습니다.

7) Try following and explain the difference from cat command.

**mycat.c**

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

void main(int argc, char \*argv[]){

   int x,y;

   char buf[20];

   x=open(argv[1], O\_RDONLY, 00777);  // open the specified file

if (x==-1){ // if there is an error

perror(“error in open”); // report it

exit(1); // and stop the program

}

   for(;;){

       y=read(x, buf, 20);                // read max 20 bytes

       if (y==0) break;                   // if end-of-file, get out

       write(1, buf, y);               // write to terminal

   }

}

Now check whether it is working similarly to “cat”.

$ cat f1

I have a dream

that one day

this nation will rise up,

live out the true meaning of its creed.

$ mycat f1

I have a dream

that one day

this nation will rise up,

live out the true meaning of its creed.

$ cat f23

cat: f23: No such file or directory

$ mycat f23

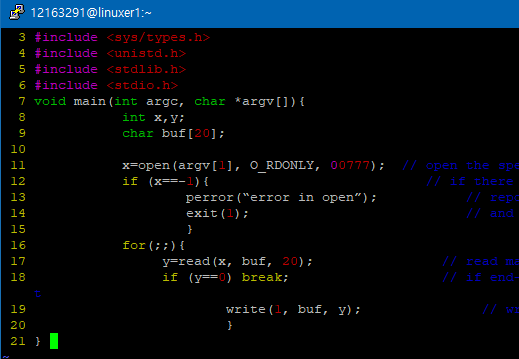
error in open: No such file or directory

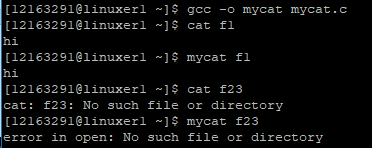
$ cat mycat.c

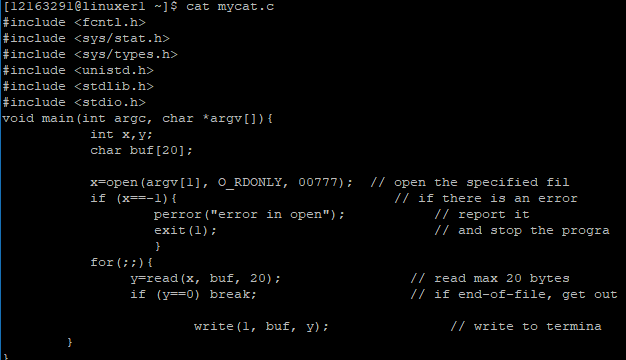
………

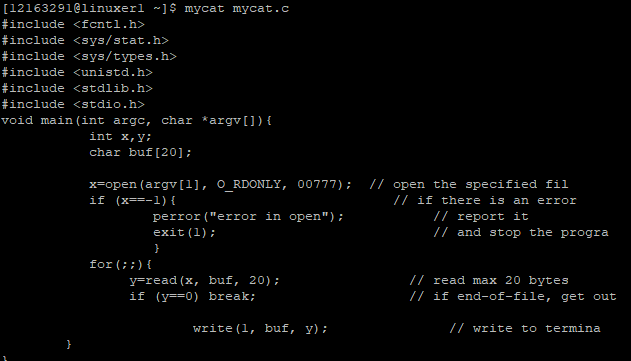
$ mycat mycat.c

……….









동일하게 동작하고 있는 것을 확인할 수 있습니다.

+ 하지만 mycat은 cat과 달리 file을 하나밖에 읽지 못합니다!



8) Try following: **mycat2.c**. Use functions for your program.

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

void display\_content(int x); // function prototype

void main(int argc, char \*argv[]){

   int x;

   x=open(argv[1], O\_RDONLY, 00777);  // open the specified file

if (x==-1){ // if there is an error

perror(“error in open”); // report it

exit(1); // and stop the program

}

   display\_content(x);

}

void display\_content(int x){

// display the content of file x in the screen

char buf[20];

int y;

for(;;){

       y=read(x, buf, 20);                // read max 20 bytes

       if (y==0) break;                   // if end-of-file, get out

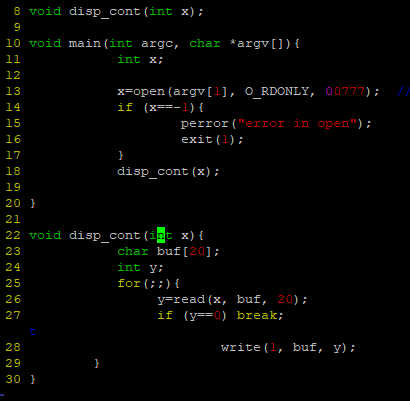
       write(1, buf, y);               // write to terminal

   }

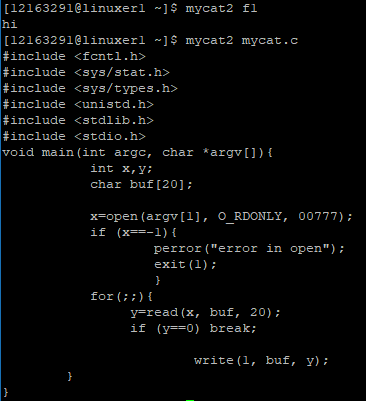
}

$ mycat2 f1

..............



7) 의 mycat 코드를 disp\_cont로 함수를 사용한 코드로 변경해 주었습니다.

 …

mycat과 동일한 결과를 냅니다.

9) Try following: **mycat3.c**.

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

void show\_file(char \* fname);

void display\_content(int x); // function prototype

void main(int argc, char \*argv[]){

show\_file(argv[1]);

}

void show\_file(char \* fname){

   int x;

   x=open(fname, O\_RDONLY, 00777);  // open the specified file

if (x==-1){ // if there is an error

perror(“error in open”); // report it

exit(1); // and stop the program

}

   display\_content(x);

}

void display\_content(int x){

// display the content of file x in the screen

char buf[20];

int y;

for(;;){

       y=read(x, buf, 20);                // read max 20 bytes

       if (y==0) break;                   // if end-of-file, get out

       write(1, buf, y);               // write to terminal

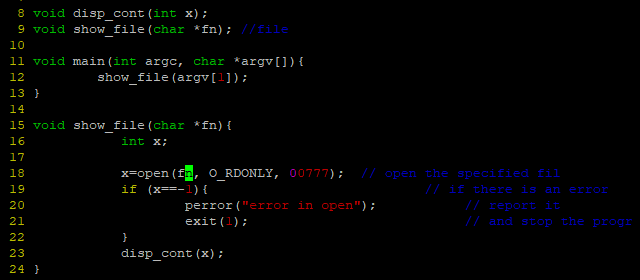
   }

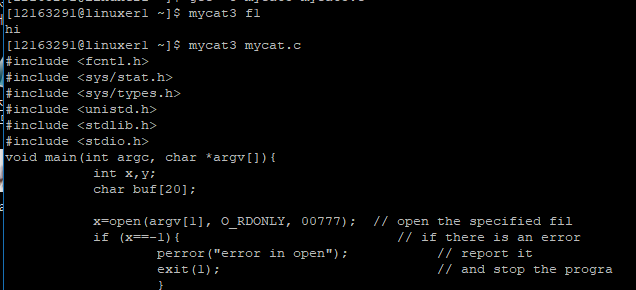
}

$ mycat3 f1

.............

9)에서 main에 남아있던 부분도 – show\_file 함수로 묶어주었습니다.





이또한 mycat 과 결과는 동일합니다.

10) You can debug programs with command line arguments as follows. Debug mycat3.c with gdb. To pass command line arguments to gdb, do "set args arg1 arg2 ...".

$gdb mycat3

gdb) b main

gdb) set args f1 f2 f3

gdb) r

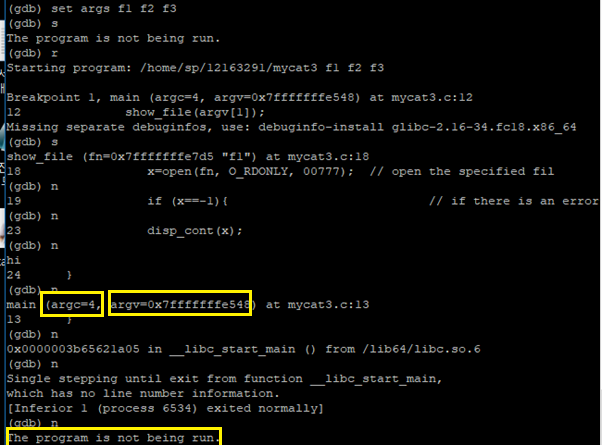
.........

gdb) s ==> execute next statement (for function, enter the function)

...

gdb) n ==> execute next statement (for function, execute whole function and return)

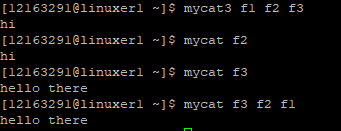
......



argc 의 값이 4이고, 주소값 하나가 main에 들어가는것을확인할 수 있습니다.

그 후 program이 끝나버립니다.

여기서, argv[1]의 값 하나만 출력되고 끝나버린다는 것을 유추할 수 있습니다.



확인하여 보니, 정말 그렇습니다.

10-1) Following program falls into infinite loop when run: ex1 f1. Debug it with gdb.

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

void main(int argc, char \*argv[]){

   int x,y;

   char buf[20];

   x=open(argv[1], O\_RDONLY, 00777);  // open the specified file

if (x==-1){ // if there is an error

perror(“error in open”); // report it

exit(1); // and stop the program

}

   for(;;){

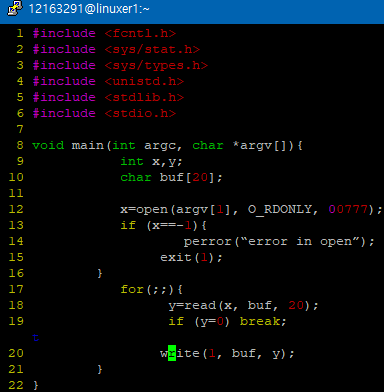
       y=read(x, buf, 20);                // read max 20 bytes

       if (y=0) break;                   // if end-of-file, get out

       write(1, buf, y);               // write to terminal

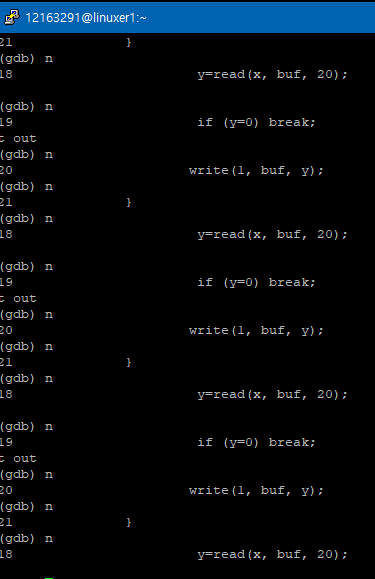
   }

}





디버그 실행합니다. arg값은 f1 . f2 입니다.



무한루프에 빠지게 된 것을 확인할 수 있었습니다

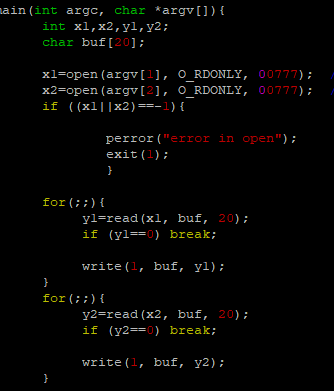
11) Modify mycat.c such that it can handle two input files.

$ mycat f1

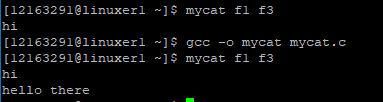
will print the contents of f1.

$ mycat f1 f2

Will print the contents of f1, and f2. The result should be same as the result of “cat f1 f2”.



똑 같은 mechanism을 두번 반복해 주었습니다.



컴파일 이후, mycat이 파일의 내용을 모두 읽는 것을 확인할 수 있었습니다.

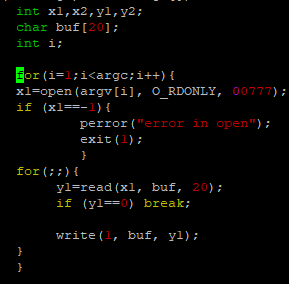
12) Modify mycat such that it can handle any number of files.

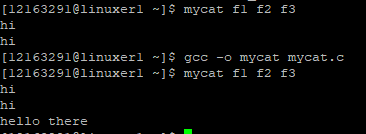
$ mycat f1 f2 f3

Will print the contents of f1, f2, and f3. The result should be same as the result of “cat f1 f2 f3”.

$ mycat f1 f2 f3 f4

will print the contents of f1, f2, f3, and f4.





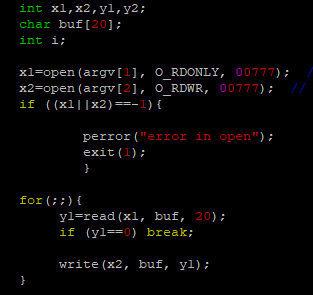
argc 값을 사용하여, 그 수만큼 for문안에서 똑같은 mechnism을 반복해주었습니다.

성공적으로 결과가 출력되었습니다.

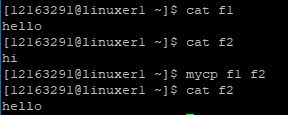
13) Implement mycp that works similarly to “cp”.

$ mycp f1 f2

will copy f1 into f2



사용된 코드는 위와 같습니다.



14) Implement myxxd that works similarly to “xxd”. Run "myxxd mycat.c". Compare the result with "xxd mycat.c".

$ cat f1

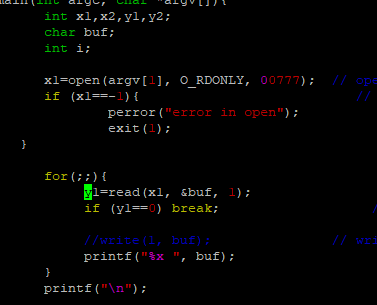
abc

$ xxd f1

0000000: 6162 630a abc.

$ myxxd f1

61 62 63 a



출력 부분에서, window에 출력할 때,

%x 형식으로 출력하면 됩니다.



f1 내용을 바꿔줍니다.



잘 출력되었습니다.

15) Modify mycat to handle various options. The second argument is either a file or an option. If it is a file, just display the contents. If it is an option (starting with ‘-‘), perform the following corresponding actions.

$ mycat -o f1 f1out

will copy f1 into f1out. (same effect as “cat f1 > f1out”)

$ mycat -x f1

will print the contents of f1 on screen in hexadecimal numbers. (similar effect as “xxd f1”)

$ mycat -p /etc/passwd

will show the contents of /etc/passwd more user-friendly as follows:

................

id: 12170099 passwd:x uid:1300 gid:1300 desc: Student Account home:/home/sp1/12170099 sh:/bin/bash

id: 12131122 passwd:x uid:1301 gid:1301 desc: Student Account home:/home/sp1/12131122 sh:/bin/bash

....................

............

You may need fopen, fgets, strtok() for this option.

You need to know the structure of /etc/passwd file with "man 5 passwd".

$ mycat -d d1

will print the name of files belonging to d1 which is a directory file.

You may need opendir(), readdir() for this option. Do "man 3 opendir", "man 3 readdir" to see the usage.

Use functions wisely.

void main(...){

......

if (strcmp(argv[1],"-o")==0){ // copy option. copy argv[2] to argv[3]

docopy(argv[2], argv[3]);

}else if (strcmp(argv[1],"-x")==0){ // xxd option

..........

}...........

........

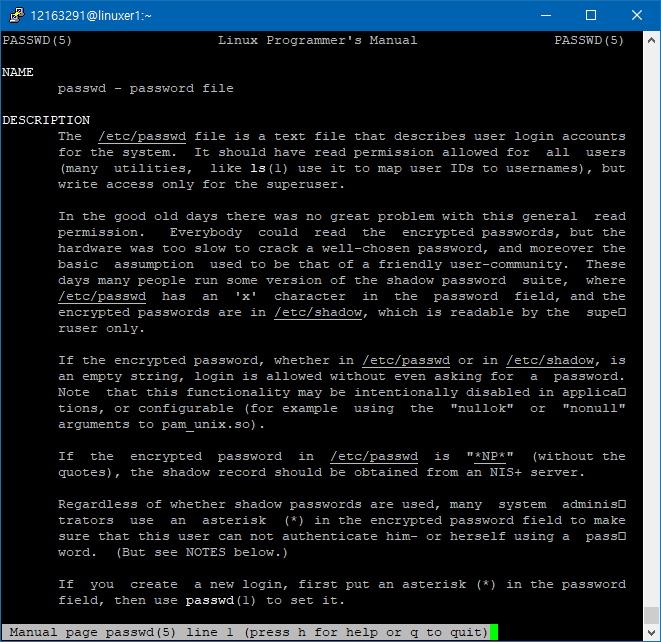
}

void docopy(char \*f1, char \*f2){ // copy f1 to f2

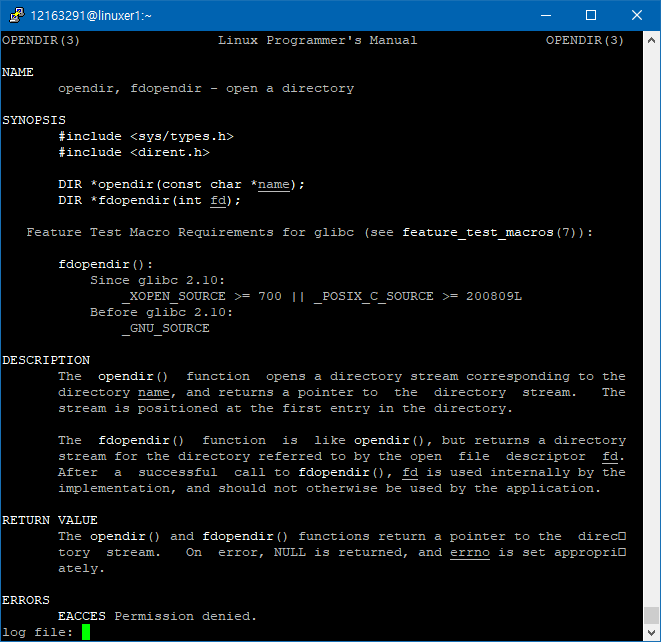
int x1 = open(f1, O\_RDONLY, 00777); // input file

int x2 = open(f2, O\_WRONLY|O\_CREAT|O\_TRUNC, 00777); // output file

.............read from x1 and store into x2 ..............



passwd



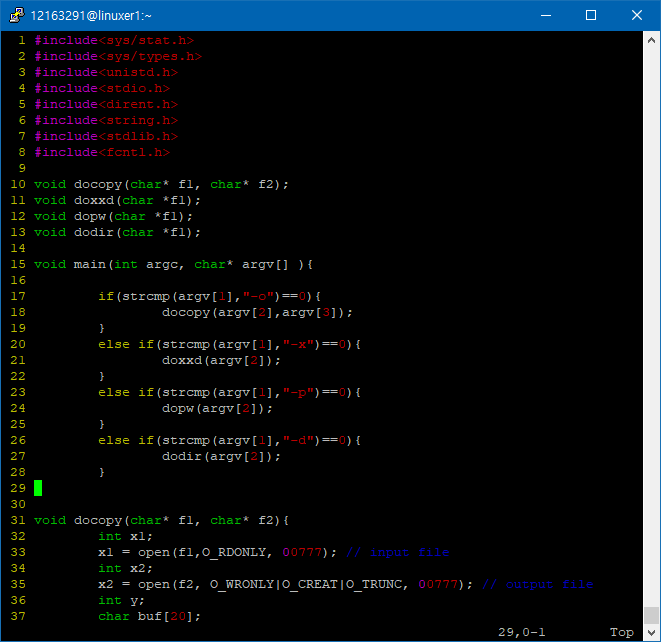
dir

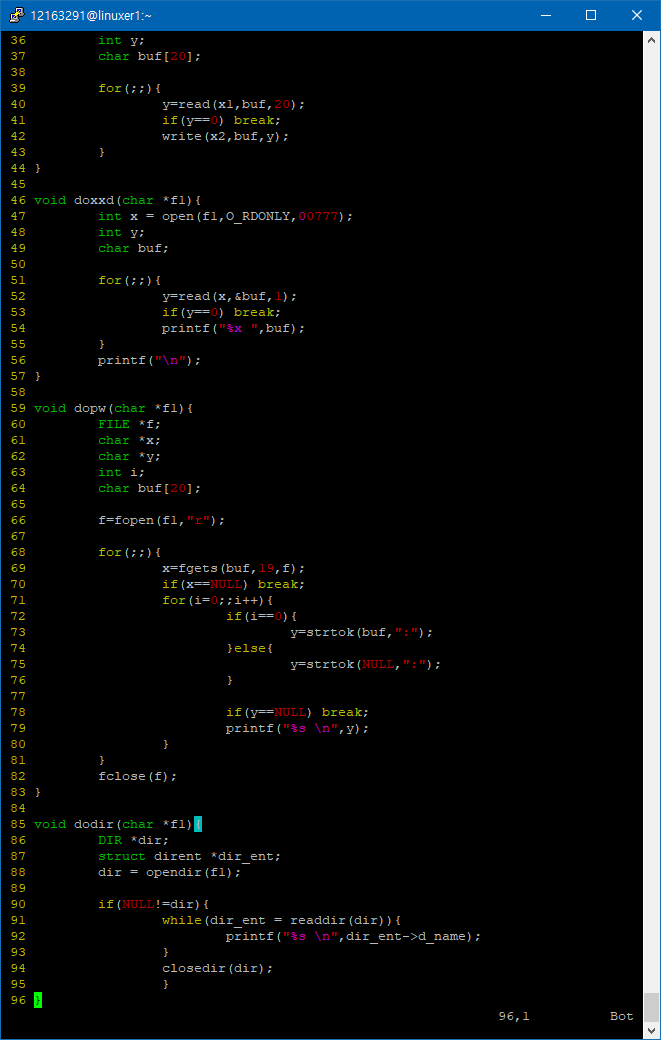


원리는 이해했는데,

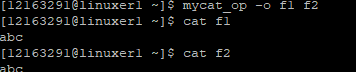
코드가 길다보니 에러가 난부분이 있는 것 같습니다.

이를 좀더 수정해 봐야할 것 같습니다.



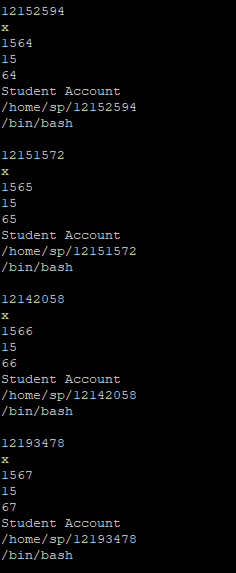


전체 코드입니다.

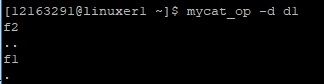


성공적으로 실행되었습니다





학생분들의 정보를 확인할 수 있었습니다.





두개의 결과가 같은 것을 확인할 수 있었습니다.