**시스템프로그래밍 lect7**

**12191833 최정희**

0) Go to the home directory with “cd ~” command. Modify shell startup files (.bash\_profile and .bashrc) so that it can add "." (current directory) in PATH environment variable. Check with “ls –a” if you have them; if you don’t, create them. For macOS, just do step 2 on “.zprofile” file in the home directory.

$ cd ~

$ ls –a

……

step 1:

Open .bash\_profile and make sure it has following lines. If not, add it.

if [ -f ~/.bashrc ]; then

source ~/.bashrc

fi

step 2:

Open .bashrc and add following line. Note you need to put “.” after “$PATH:”

export PATH=$PATH:.

**And close your terminal and reopen**. Now you can move to any directory and type a program name without "./" prefix.

But first check if PATH environment variable contains “.” at the end.

$ echo $PATH

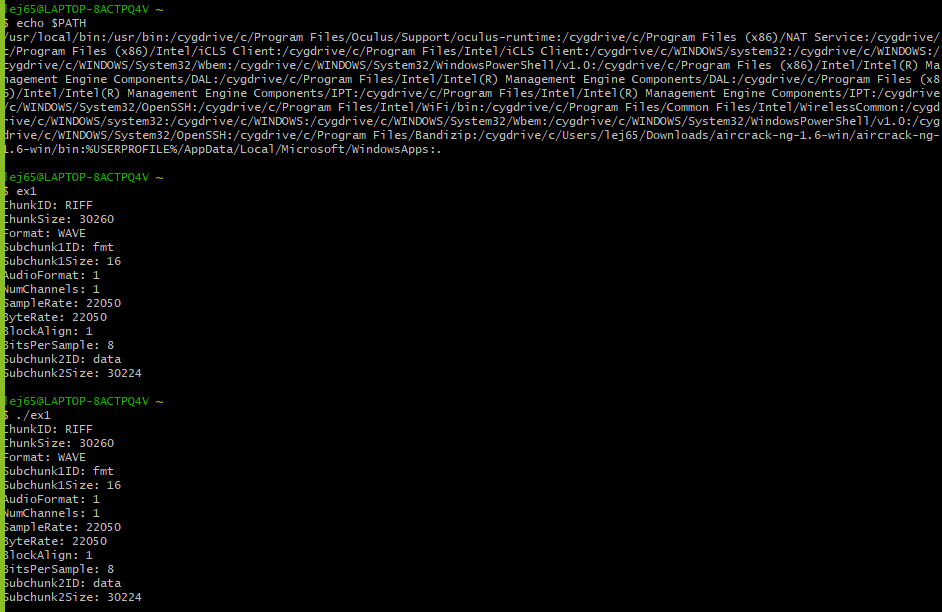
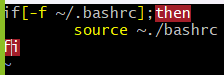
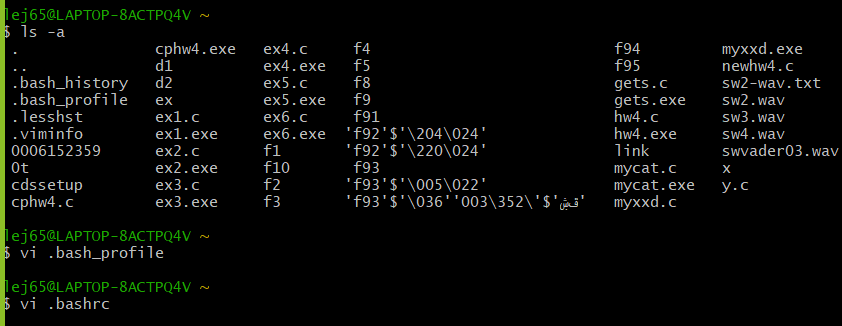
………………….

And try

$ ex1

instead of

$./ex1



조건문으로 ~/.bashrc파일이 존재하는지 확인하고, 해당파일이 존재할 경우 source 명령어를 사용하여 해당 파일을 실행합니다. Export PATH=$PATH:.는 환경변수인 PATH에 현재 디렉토리를 추가합니다. 결과적으로 현재 디렉토리를 PATH에 추가하여 현재 디렉토리에서 실행 가능한 프로그램을 실행할 때는 ./ex1 대신 ex1을 하여도 실행이 되게 만듭니다.

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

**ex0.c**:

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

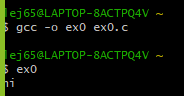
printf("hi\n");

}

$ gcc –o ex0 ex0.c

$ ex0

hi



Ex0.c는 argument를 받지 않고 실행하는 프로그램입니다.

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

}

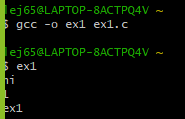
$ gcc –o ex1 ex1.c

$ ex1

hi

1

ex1



Void main()의 괄호안에 int형과 char 포인터형 argument를 추가함으로 ex1을 실행했을 때 argv[0]은 ex1이 되고 argc는 argument가 1개 이므로 1을 출력하게 됩니다.

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

}



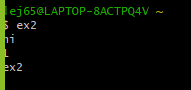
$ gcc –o ex2 ex2.c

$ ex2

hi

1

ex2



Segmentation fault (core dumped)

=> You have to provide two command line arguments!

Argv[1]까지 printf를 했지만 argv[1]이 없기 때문에 segmentation fault가 생겼습니다.

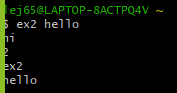
$ ex2 hello

hi

2

ex2

hello



Argv[1]에 hello를 입력하여 argc는 2개 argv[2]인 hello가 출력됩니다.

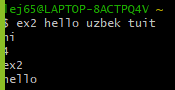
$ ex2 hello uzbek tuit

hi

4

ex2

hello



Ex2.c에서 argv[1]까지만 출력되게 했으므로 그 이상의 argument를 입력하면 출력되지 않습니다. 프로그램이 받을 수 있는 argument보다 많이 받게 되면 나머지는 무시할 수 있습니다.

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

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

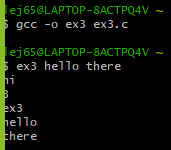
hi

3

ex3

hello

there



Ex3.c는 3개의 argument를 출력할 수 있는 프로그램 입니다 argv[0]은 ex3, argv[1]은 hello, argv[2]는 there입니다

5) A program that receives any number of arguments.

**ex4.c**:

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

int i;

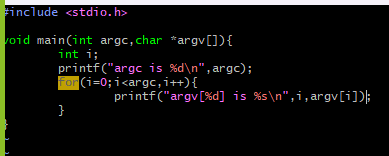
pirntf("argc is %d\n", argc);

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

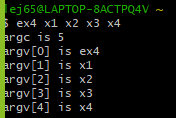
argv[0] is ex4

argv[1] is x1

argv[2] is x2

argv[3] is x3

argv[4] is x4



Ex4는 몇 개의 argument인지 상관없이 받을 수 있는 프로그램입니다. for문을 이용하여 argument가 쓰여지면 argc에 저장되어 그 수에 해당하는 만큼 argv를 출력할 수 있습니다.

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

**myecho.c**

#include <stdio.h>

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

int i;

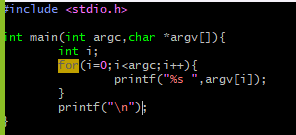
for(i=1;i<argc;i++){ // skip program name

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

}

printf("\n");

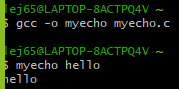
}



$ gcc –o myecho myecho.c

$ myecho hello

Hello



$ echo hello

Hello



$ myecho hello hi bye

hello hi bye



$ echo hello hi bye

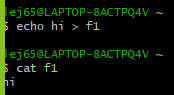
hello hi bye



$ echo hi > f1

$ cat f1

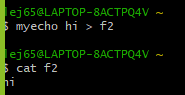
Hi



$ myecho hi > f2

$ cat f2

hi



Myecho는 echo와 비슷하게 동작하지만 옵션을 수행하지 못합니다. Echo hi > f1을 하면 f1에 hi가 저장되지만, myecho hi > f2를 하면 hi가 f2에 저장되지않습니다. 위의 결과는 f2에 이미 hi가 있었기 때문에 hi가 출력된 모습입니다.

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>

int 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(;;){

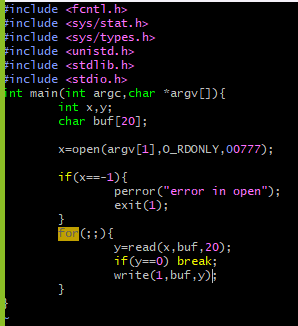
       y=read(x, buf, 20);                // read max 20 bytes

       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”.

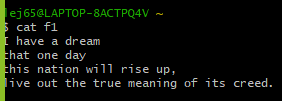
$ cat f1

I have a dream

that one day

this nation will rise up,

live out the true meaning of its creed.



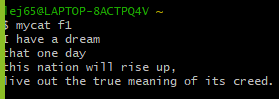
$ mycat f1

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



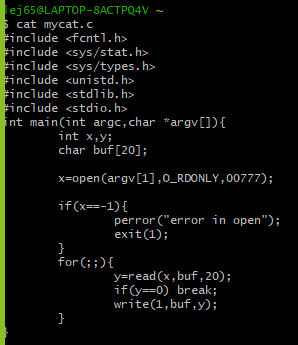
$ mycat f23

error in open: No such file or directory



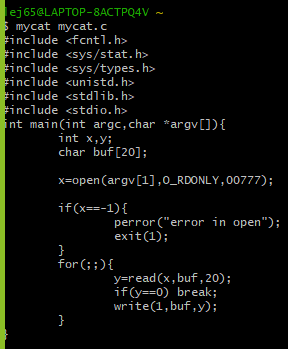
$ cat mycat.c

………



$ mycat mycat.c

……….

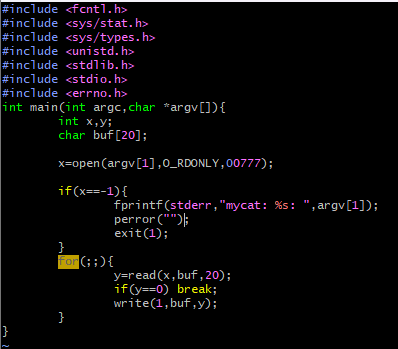


Cat과 mycat은 비슷한 동작을 하지만 파일이 존재하지 않는 경우 error메시지가 다르게 됩니다. Mycat은 perror을 사용하여 open in error라는 내용을 내보내고 에러메세지가 뜹니다. 하지만 리눅스의 에러메세지는 파일이름 : 에러메세지가 뜨는 것을 볼 수 있습니다.

7-1) Modify mycat.c such that it produces almost the same output as cat when there is an error.

$ mycat f23

mycat: f23: No such file or directory



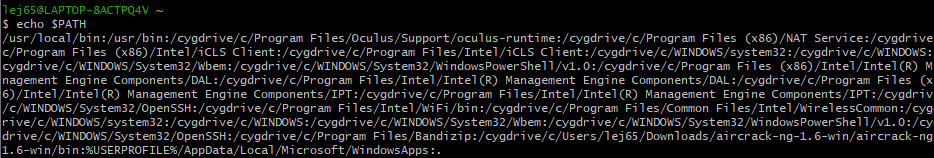
Perror은 argument를 받을 수 없습니다. 때문에 errno.h를 추가하고 fprintf로 에러메세지에 argument를 받을 수 있도록 구현했습니다.

7-2) Modify myecho.c such that it can handle environment variables. Use getenv() for this purpose. **Use ‘%’ instead of ‘$’ to denote an environment variable** to avoid shell expansion. Shell expands $STR to STR’s value when STR is an environment variable. For example, $PATH will be expanded to the value of PATH environment variable.



$ echo $PATH -- shell expands $PATH to its value and echo will print them

-- as a result, this will print the value of environment variable PATH

…………….. 

$ echo %PATH -- shell does not expand %PATH to the value of PATH and

-- echo will simply print “%PATH”

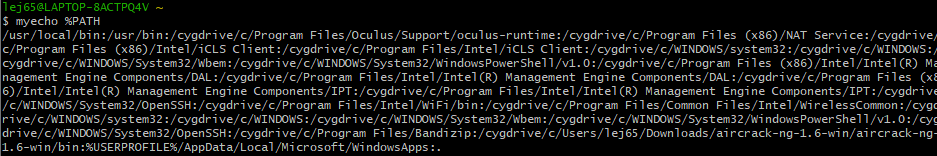
%PATH



$ myecho %PATH -- myecho recognizes %PATH as an environment variable

and shows the value of environment variable PATH

……………



$ myecho %HOME -- myecho will print the value of environment variable HOME

…….



Echo는 환경 변수로 %는 받아들이지 못하고, $만 받아들입니다. 하지만 myecho에 name이 가리키는 문자열과 일치하는 문자열을 찾기 위해 환경변수 리스트를 탐색하는 getenv를 사용해서 %도 환경 변수로 받을 수 있게 했습니다. 따라서 myecho %PATH를 했을 때 환경변수인 PATH가 나온 것을 볼 수 있습니다.

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

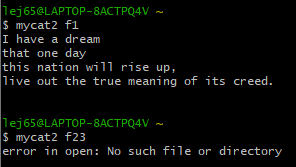
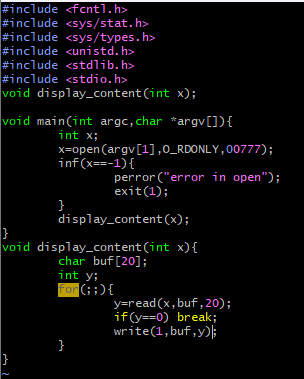
       write(1, buf, y);               // write to terminal

   }

}

$ mycat2 f1

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



Mycat2.c는 display\_content라는 파일의 내용을 보여주는 함수를 따로 구현하는 방법을 썼습니다.

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(;;){

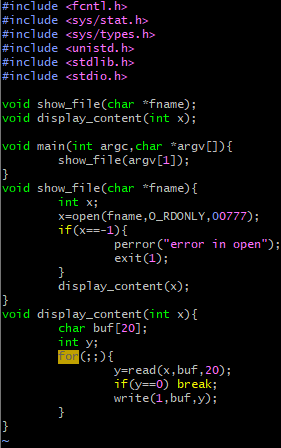
       y=read(x, buf, 20);                // read max 20 bytes

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

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

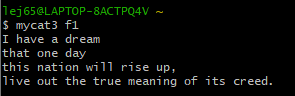
   }

}



$ mycat3 f1

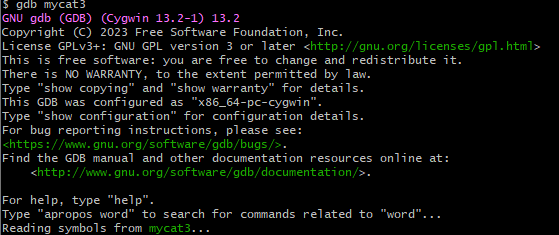
.............



Mycat3.c는 show\_file과 display\_content함수를 구현하여 show\_file은 file을 열고, display\_content는 읽고 써서 파일의 내용을 출력 시킵니다.

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



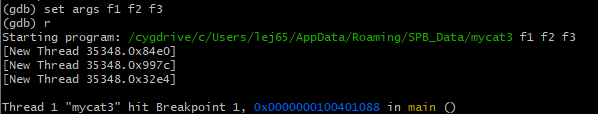
gdb) b main



gdb) set args f1 f2 f3

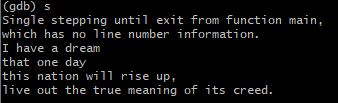
gdb) r

.........



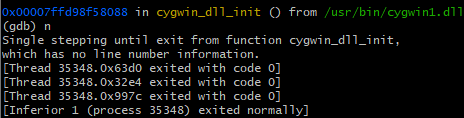
gdb) s ==> execute next statement (for function, enter the function)

...



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

......



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(;;){

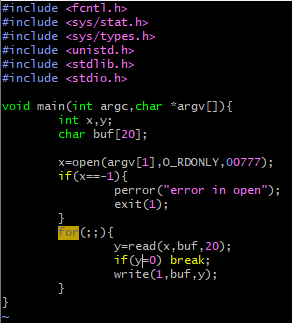
       y=read(x, buf, 20);                // read max 20 bytes

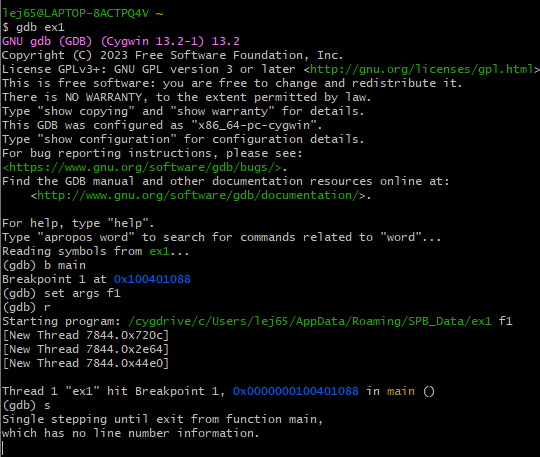
       if (y=0) break;                   // if end-of-file, get out

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

   }

}





for문의 if가 y=0이기 때문에 제대로 실행 되지않는 것을 볼 수 있습니다. Y==0으로 바꿔주면 제대로 실행이 됩니다.

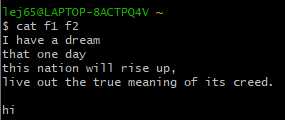
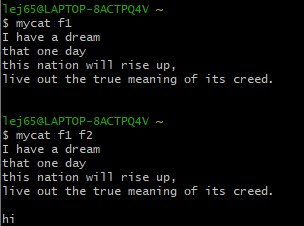
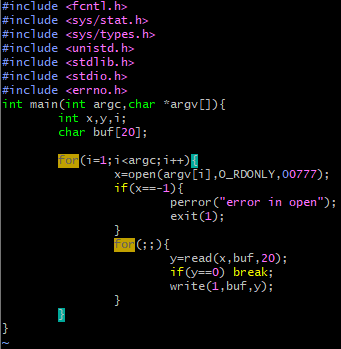
11) Modify mycat.c such that it can handle two input files.

$ 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”.



for문으로 argc만큼의 파일을 출력할 수 있는 프로그램을 구현했습니다. 위의 출력을 보면 mycat f1 f2와 cat f1 f2가 같은 결과를 나타내는 것을 볼 수 있습니다.

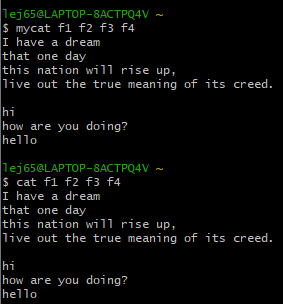
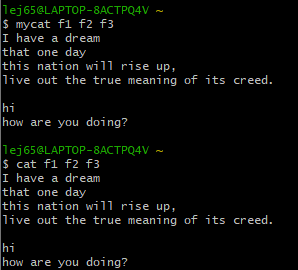
12) Modify mycat such that it can handle any number of files.

$ 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.

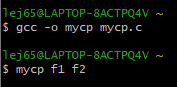
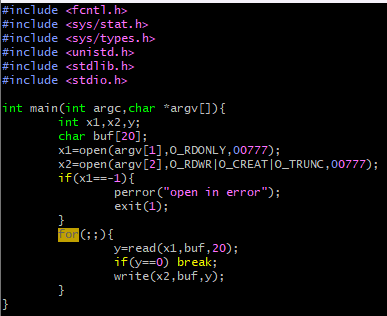


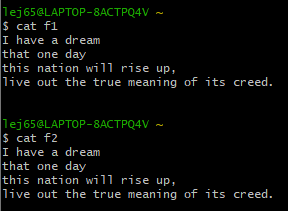
11번 문제를 구현할 때 몇 개의 argument가 입력 되더라도, 파일 내용을 전부 출력 시키도록 만들었으므로, 11번 문제의 코드를 그대로 썼습니다. Mycat f1 f2 f3 f4를 보면 몇 개의 argument가 입력되도 모두 출력하는 것을 볼 수 있습니다.

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

$ mycp f1 f2

will copy f1 into f2





Mycp는 cp와 동일한 역할을 하도록 구현했습니다. 먼저 복사할 파일은 O\_RDONLY로 읽기 전용으로 쓰고 복사 받을 파일은 cat f1 f2처럼 f2가 존재하지 않아도 f2를 만들고 복사를 할 수 있도록 O\_RDWR| O\_CREAT| O\_TRUNC로 구현했으며, 복사할 파일의 내용 read해 복사 받을 파일에 write하도록 만들었습니다.

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

$ cat f1

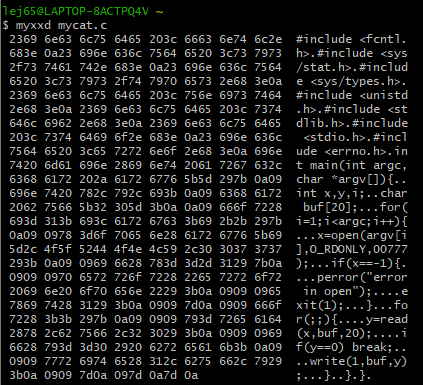
abc

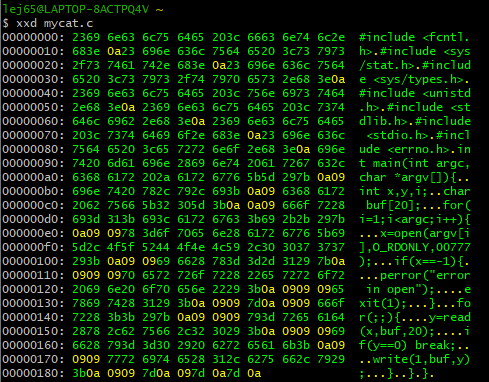
$ xxd f1

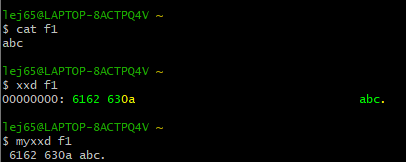
0000000: 6162 630a abc.

$ myxxd f1

61 62 63 a







Myxxd.c는 xxd와 동일한 동작을 하도록 만들었습니다. 먼저 x에 있는 내용을 buf에 16바이트 씩 읽었습니다. 그 후 제가 만든 myxxd에서는 buf[i]가 16보다 작으면 0a이런 식이 아니라 a만 출력하기 때문에, if문으로 0a로 출력하도록 만들었고, xxd를 했을 때 수평 tap과 개행은 .으로 표시되는 것을 볼 수 있었습니다. 때문에 16진수를 표현한후 buf[i]==10,9를 buf[i]=46으로 해줘 .으로 만들었습니다. Ascii table에서 십진수로 46은 .을 의미합니다. 그 후 byte를 +=1해주어 byte가 2로 나누어 떨어지면 스페이스바를 16으로 나누어 떨어지면 문자열을 출력하고 엔터를 치게 만들었습니다. 여기까지 구현했을 때, 마지막 줄의 문자열이 출력되지 않는 문제가 있었기 때문에 byte가 16으로 나누어 떨어지지 않을 때 곧 마지막 buf의 내용을 출력하도록 만들었습니다. Myxxd mycat.c와 myxxd f1을 했을 때 모두 동일한 결과가 나오는 것을 확인할 수 있습니다.

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

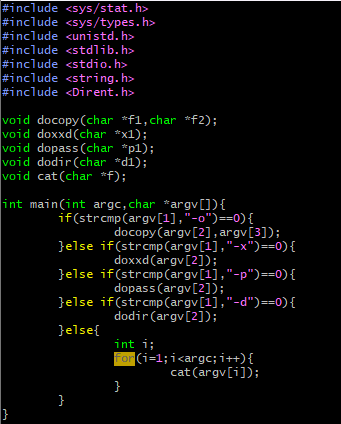
int x1 = open(f1, O\_RDONLY, 00777); // input file

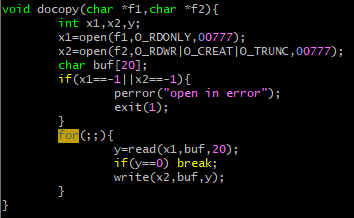
int x2 = open(f2, O\_WRONLY|O\_CREAT|O\_TRUNC, 00777); // output file

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

}

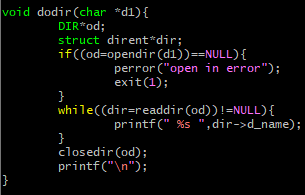
................

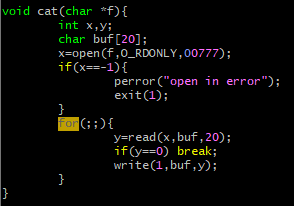




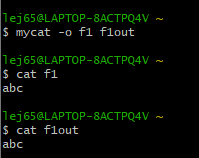




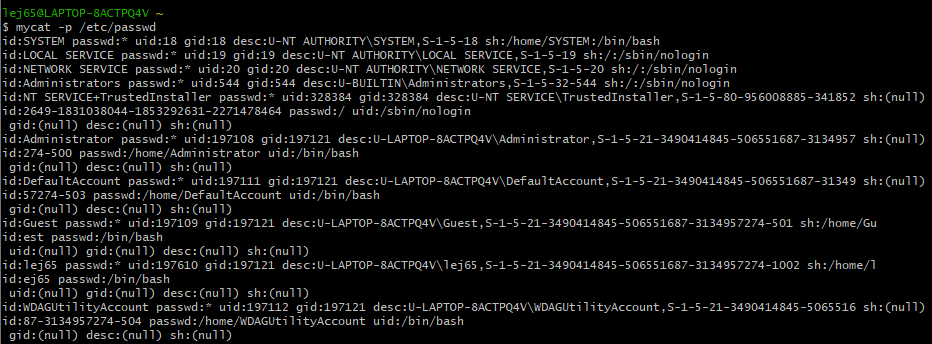


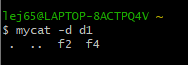


옵션을 받으면 그 옵션에 대한 것을 수행할 수 있도록 main함수에 strcmp를 이용해 argv[1]의 문자가 -o ,-x, -p ,-d에 따라 docopy,doxxd,dopass,dodir,cat함수를 만들었습니다. Docopy와 doxxd 와 cat은 전의 문제와 똑같이 구현했습니다 dopass는 사용자의 id, password, uid, gid, desc, sh를 나타내는 함수 입니다. 예전에는 학교에서 서버를 이용하는 putty를 사용한 것으로 알고 있기 때문에 모든 학생들의 id같은 것이 나왔을텐데, 저는 시그윈을 이용한 터미널을 사용하기 때문에 그런건 나오지 않았습니다. Dodir함수는 opendir로 directory에 접근하여 directory가 없으면 perror로 open in error를 반환하고 directory가 존재하면 dir->name으로 directory에 있는 파일 이름을 출력하도록 만들고 출력이 끝나면 closedir로 directory를 닫았습니다. Opendir과 closedir을 이용하기 위해 Dirent.h를 추가했습니다.

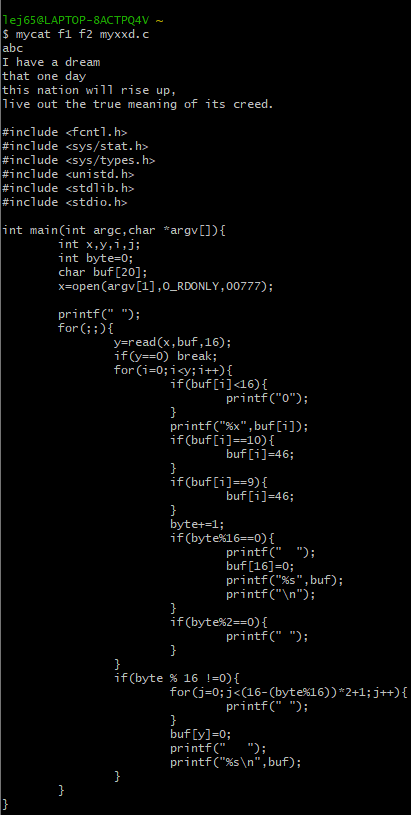












출력 결과를 보면 mycat -o f1 f1out을 한 결과 cat f1 > f1out과 동일한 결과를 나타내고 mycat -x f2를 한 결과 xxd f2와 동일한 결과를 나타내고 mycat -p /etc/passwd를 한 결과 id,passwd,uid,gid,desc,sh를 나타내는 것을 볼 수 있고, mycat -d d1을 한결과 d1에 있는 파일을 출력하는 것을 d1 directory에서 ls -a 커맨드를 입력하여 확인하였습니다. 아무 옵션도 쓰지 않으면 단순히 출력만 합니다.