8. Homework

12201922 이규민

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

1) Try following program which doesn't receive command line arguments.
ex0.c:

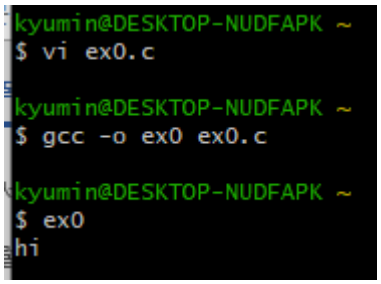
```
void main(){   // this program doesn't receive command line arguments
    printf("hi₩n");
```

}

$ gcc –o ex0 ex0.c

$ ex0

Hi



./ex0가 아닌 ex0만 입력해도 프로그램이 실행된다. 강의 노트에 나와있는대로 진행 후 터미널을 껐다가 다시 켜면 "./" 없이 바로 이름만 입력하면 된다.

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

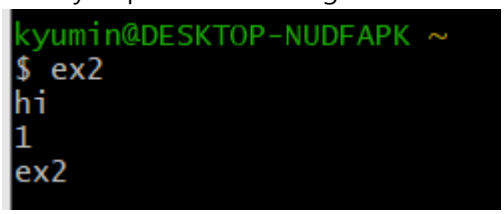
Ex1이라는 하나의 argument가 입력되었기 때문에 개수는 1과 ex1 메시지가 출력되었다.

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

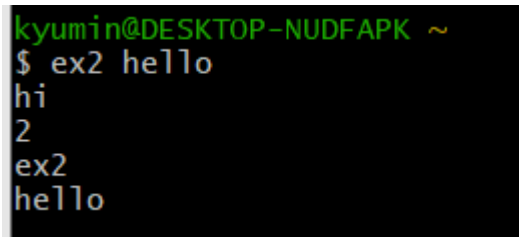
ex2.c:

```
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
```

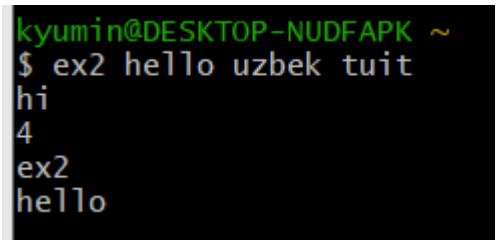
}
$ gcc –o ex2 ex2.c
$ ex2
hi
1
ex2
Segmentation fault (core dumped)
=> You have to provide two command line arguments!
$ ex2 hello
hi
2
ex2
hello
$ ex2 hello uzbek tuit
hi
4
ex2
hello
=> If you provide more arguments than what the program can receive, the rest will be ignored.



코드에는 argc, argv[0], argv[1]을 출력하도록 만들어져있지만 입력된 Argument가 하나이므로 ex2 메시지만 입력되었다. Argv[1]은 아무것도 출력되지 않는다.



Ex2와 hello라는 두 개의 argument가 입력되었다. 그래서 ex2와 hello모두 출력된다.

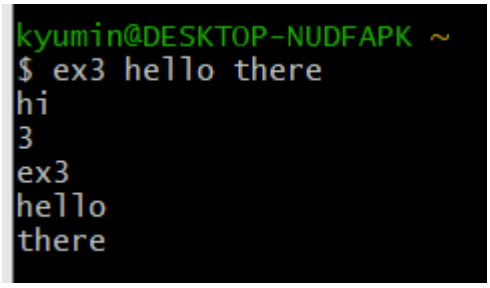


4개의 argument가 입력되었지만 코드에서는 첫 번째와 두번째의 argument만 출력되도록 만들었기 때문에 uzbek tuit은 출력되지 않는다.

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
```



3개의 argument를 출력하는 코드를 작성하였고 정상 작동하는 것을 볼 수 있다.

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.

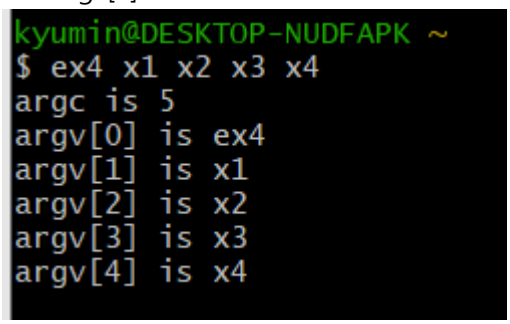
```
$ ex4 x1 x2 x3 x4
argc is 5
argv[0] is ex4
```

argv[1] is x1

argv[2] is x2

argv[3] is x3

argv[4] is x4



위 코드는 입력 받은 argument의 수만큼 for문을 반복하여 argument를 출력하는 코드다. 예제에서는 5개의 argument를 입력했고, 모두 출력된 것을 볼 수 있다.

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

```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o myecho myecho.c

kyumin@DESKTOP-NUDFAPK ~
$ myecho hello
hello

kyumin@DESKTOP-NUDFAPK ~
$ echo hello
hello

kyumin@DESKTOP-NUDFAPK ~
$ myecho hello hi bye
hello hi bye

kyumin@DESKTOP-NUDFAPK ~
$ echo hello hi bye
hello hi bye
```

```
kyumin@DESKTOP-NUDFAPK ~
$ echo hi > f1

kyumin@DESKTOP-NUDFAPK ~
$ cat f1
hi

kyumin@DESKTOP-NUDFAPK ~
$ myecho hi > f2

kyumin@DESKTOP-NUDFAPK ~
$ cat f2
hi
```

위 코드는 입력 받은 argument 중에서 첫번째 만을 제외하고 나머지 argument를 출력하는 코드다. 즉 echo 명령어와 동일하게 작동한다. myecho hi > f2 라고 입력하면 hi라는 내용이 저장된 f2 파일을 생성할 수 있다.

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

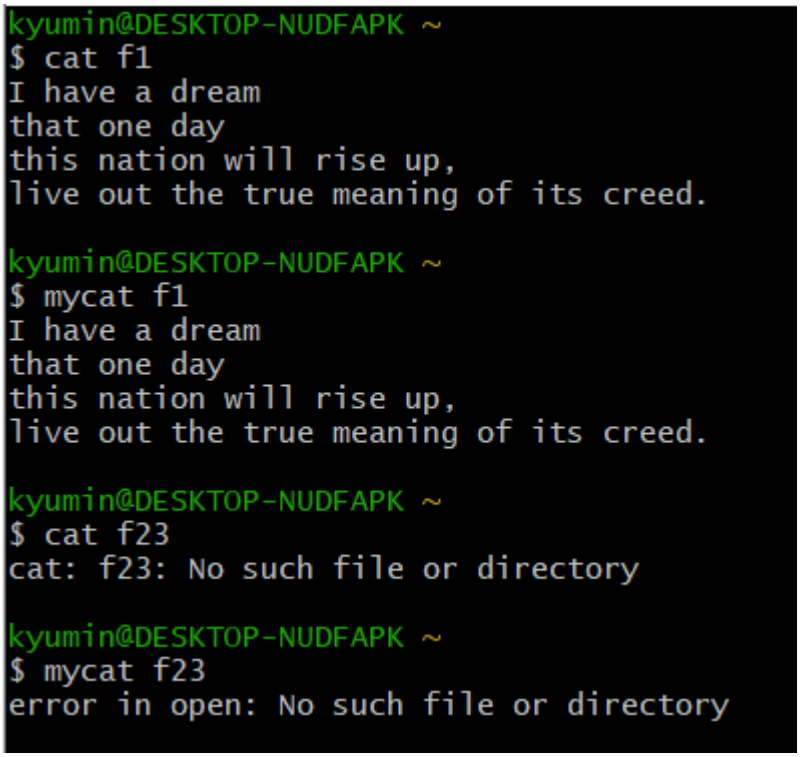
mycat.c

```
#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
…….



```
kyumin@DESKTOP-NUDFAPK ~
$ cat 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
   }
```

```
kyumin@DESKTOP-NUDFAPK ~
$ mycat 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(;;){
```

위 코드는 프로그램 다음으로 입력된 argument 의 내용을 화면에 출력하는 코드이다. 즉 cat 과 동일하게 작동한다. 그래서 mycat mycat.c 를 입력하면 mycat.c 의 파일 내용을 읽어서 화면에 출력한다.

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

```c
int main(int argc, char *argv[]){
    int x,y;
    char buf[20];

    x=open(argv[1], O_RDONLY, 00777);  // open
    if (x==-1){                          // i
        perror("mycat");            // report i
        exit(1);                             //
    }
    for(;;){
        y=read(x, buf, 20);                  //
        if (y==0) break;                     //
        write(1, buf, y);                  // wri
    }
```

```
kyumin@DESKTOP-NUDFAPK ~
$ mycat f23
mycat: No such file or directory
```

파일을 오픈하지 못 한 경우에 출력 메시지만 바꾸면 된다. Perror("mycat");으로 수정하니 원하는 결과가 출력된 것을 볼 수 있다.

7-2) Modify myecho.c such that it can handle environment variables. Use getenv() for this purpose.

$ echo $PATH -- this will print the value of environment variable PATH
……..
$ myecho $PATH -- myecho should show the value of environment variable PATH
……..

```
kyumin@DESKTOP-NUDFAPK ~
$ echo $PATH
/usr/local/bin:/usr/bin:/cygdrive/c/WINDOWS/system32:/cygdrive/c/WINDOWS:/cygdri
ve/c/WINDOWS/System32/Wbem:/cygdrive/c/WINDOWS/System32/WindowsPowerShell/v1.0:/
cygdrive/c/WINDOWS/System32/OpenSSH:/cygdrive/c/Program Files/Git/cmd:/cygdrive/
c/Program Files/dotnet:/cygdrive/c/Users/kyumin/AppData/Local/Programs/Python/Py
thon37/Scripts:/cygdrive/c/Users/kyumin/AppData/Local/Programs/Python/Python37:/
cygdrive/c/Users/kyumin/AppData/Local/Microsoft/WindowsApps:.

kyumin@DESKTOP-NUDFAPK ~
$ myehco $PATH
-bash: myehco: command not found
```

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

```c
#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
.............
```

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
.............
```

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
    }
}
```

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".

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.

13) Implement mycp that works similarly to "cp".

$ mycp f1 f2

will copy f1 into f2

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

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 ..............
}
.................
```