

## 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.

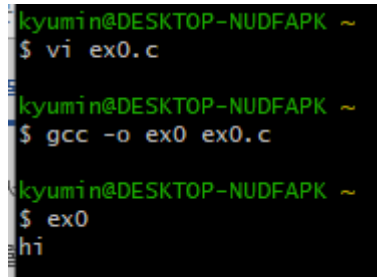
**ex0.c:**

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

```
$ gcc -o ex0 ex0.c
```

```
$ ex0
```

```
Hi
```

A terminal window with a black background and green text. The prompt is 'kyumin@DESKTOP-NUDFAPK ~'. The user enters '\$ vi ex0.c', then '\$ gcc -o ex0 ex0.c', and finally '\$ ex0'. The output of the program is 'hi'.

./ex0가 아닌 ex0만 입력해도 프로그램이 실행된다. 강의 노트에 나와있는대로 진행 후 터미널을 꺾다가 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
```

```
}
```

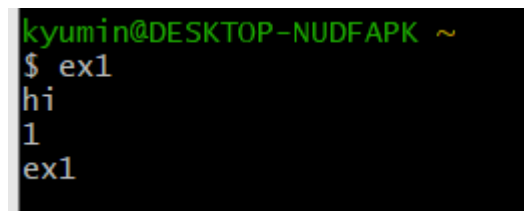
```
$ gcc -o ex1 ex1.c
```

```
$ ex1
```

```
hi
```

```
1
```

```
ex1
```

A terminal window with a black background and green text. The prompt is 'kyumin@DESKTOP-NUDFAPK ~'. The user enters '\$ ex1'. The output of the program is 'hi', '1', and 'ex1' on separate lines.

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
```

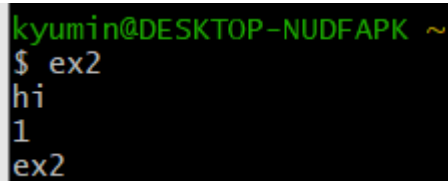
```
hi
```

```
4
```

```
ex2
```

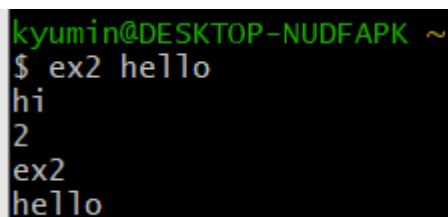
```
hello
```

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



```
kyumin@DESKTOP-NUDFAPK ~
$ ex2
hi
1
ex2
```

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



```
kyumin@DESKTOP-NUDFAPK ~
$ ex2 hello
hi
2
ex2
hello
```

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

```
kyumin@DESKTOP-NUDFAPK ~  
$ ex2 hello uzbek tuit  
hi  
4  
ex2  
hello
```

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
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ex3 hello there  
hi  
3  
ex3  
hello  
there
```

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

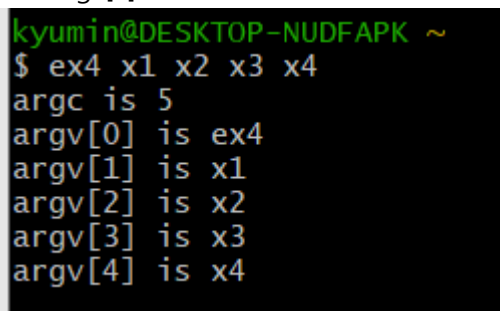
5) A program that receives any number of arguments.

**ex4.c:**

```
void main(int argc, char *argv[]){
    int i;
    printf("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
```

A terminal window with a black background and green text. The prompt is 'kyumin@DESKTOP-NUDFAPK ~'. The command '\$ ex4 x1 x2 x3 x4' has been entered. The output shows 'argc is 5' followed by five lines of 'argv[i] is [argument]', where i ranges from 0 to 4. The arguments are 'ex4', 'x1', 'x2', 'x3', and 'x4' respectively.

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

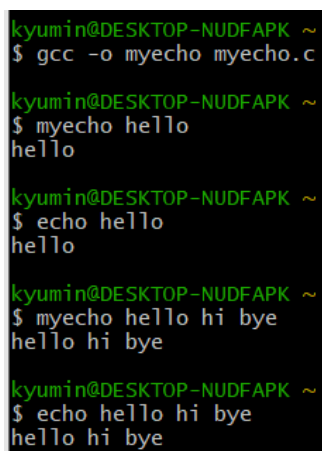
```
$ cat f1
```

```
hi
```

```
$ myecho hi > f2
```

```
$ cat f2
```

```
Hi
```



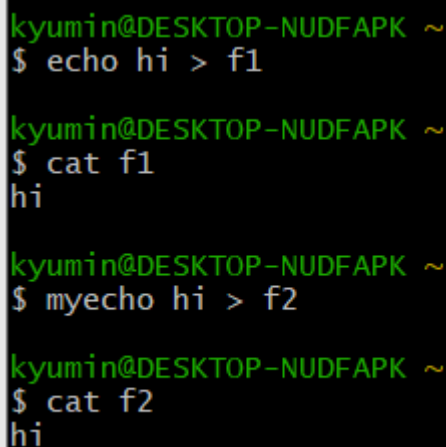
```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o myecho myecho.c

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 f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.

kyumin@DESKTOP-NUDFAPK ~
$ mycat f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.

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

kyumin@DESKTOP-NUDFAPK ~
$ mycat f23
error in open: No such file or directory

```

```

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
```

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

```

#include <stdio.h>
#include <stdlib.h>
void main(int argc, char *argv[]){
    //두 개의 argument가 들어오고 두 번째 문자열이 %로 시작하는 경우
    if(argc==2 && argv[1][0]=='%'){
        //환경 변수 출력하기
        if(getenv(argv[1] + 1) != NULL){
            printf("%s", getenv(argv[1] + 1));
            exit(1);
        }
        else printf("환경 변수가 존재하지 않습니다.");
        exit(0);
    }

    //환경 변수가 입력되지 않은 경우 입력된 메시지 출력
    for(int i=1; i<argc; i++){ // skip program name
        printf("%s ", argv[i]); // and display all the arguments
    }
    printf("\n");
}

```

우선 argument 가 입력된 개수가 2 개인지 확인하고, 두 번째 argument 의 첫번째 글자가 %인지 확인 하였다.(일반적으로 \$을 입력해야하지만, 다른 학생들 중 코드를 구현하지 않아도 환경변수 확인이 가능한 문제가 있어 %로 구현했음.)

만약 그렇다면 입력된 환경변수를 getenv 함수에 넣어 출력을 해야하지만 입력된 argument 앞 글자에는 %라는 문자가 있다. 그러므로 getenv(argv[1] + 1)을 출력한다면 %를 제외한 문자열만 입력이 된다. 그 이유는 배열에는 주소가 있는데, 그 주소에는 배열의 첫번째 문자가 들어있다. 그리고 주소에 1 을 더한 값인 그 다음 주소에는 배열의 두 번째 글자가 들어있다. 그래서 주소를 입력할 때 argv[1]이 아닌 argv[1] + 1 을 입력하면 두 번째 글자부터 입력이 되는 것이다.

만약 if 문을 만족하지 않는다면, 즉 환경변수 입력이 되지 않았다면 입력된 문자열만 화면에 출력하고 프로그램을 종료한다.

```

kyumin@DESKTOP-NUDFAPK ~
$ echo $PATH
/usr/local/bin:/usr/bin:/cygdrive/c/WINDOWS/system32:/cygdrive/c/WINDOWS:/cygdrive/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/Program Files/PuTTY:/cygdrive/c/Users/kyumin/AppData/Local/Programs/Python/Python37/Scripts:/cygdrive/c/Users/kyumin/AppData/Local/Programs/Python/Python37:/cygdrive/c/Users/kyumin/AppData/Local/Microsoft/WindowsApps:.

```

```

kyumin@DESKTOP-NUDFAPK ~
$ myecho %PATH
/usr/local/bin:/usr/bin:/cygdrive/c/WINDOWS/system32:/cygdrive/c/WINDOWS:/cygdrive/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/Program Files/PuTTY:/cygdrive/c/Users/kyumin/AppData/Local/Programs/Python/Python37/Scripts:/cygdrive/c/Users/kyumin/AppData/Local/Programs/Python/Python37:/cygdrive/c/Users/kyumin/AppData/Local/Microsoft/WindowsApps:.

```

```

kyumin@DESKTOP-NUDFAPK ~
$ echo $HOME
/cygdrive/c/Users/kyumin/AppData/Roaming/SPB_Data

kyumin@DESKTOP-NUDFAPK ~
$ myecho %HOME
/cygdrive/c/Users/kyumin/AppData/Roaming/SPB_Data

```

Echo \$PATH 와 myecho %PATH 와 동일한 결과가 나오는 것을 볼 수 있다.

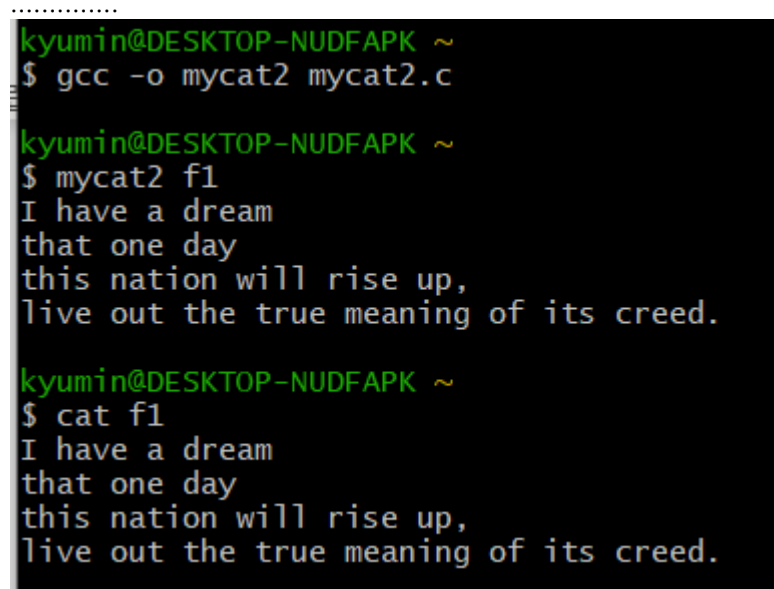
8) Try following: mycat2.c. Use functions for your program.

```
#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



```
.....
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o mycat2 mycat2.c

kyumin@DESKTOP-NUDFAPK ~
$ mycat2 f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.

kyumin@DESKTOP-NUDFAPK ~
$ cat f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.
```

문제에 주어진 코드를 실행해보면 cat 과 동일하게 작동하는 것을 볼 수 있다. 이전에 만들었던 mycat 과도 동일하게 작동한다. Mycat2 가 Mycat 과 다른점은 입력된 파일의 내용을 출력할 때 함수를 통해서 출력했다는 점이다.

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>

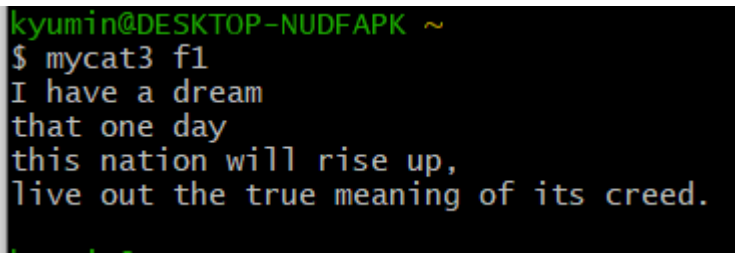
void show_file(char * fname);
void display_content(int x); // function prototype

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

A terminal window with a black background and green text. The prompt is 'kyumin@DESKTOP-NUDFAPK ~'. The user enters '\$ mycat3 f1'. The output is 'I have a dream', 'that one day', 'this nation will rise up,', and 'live out the true meaning of its creed.' on separate lines.

```
kyumin@DESKTOP-NUDFAPK ~
$ mycat3 f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.
```

Mycat3 은 cat 명령어와 동일하게 작동한다. Mycat3.c 의 코드에서는 파일을 읽고 화면에 출력하는 과정 모두를 함수를 통해 진행되도록 만들었다.

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)
```

```
.....
```

```
(gdb) set args f1 f2 f3
(gdb) r
Starting program: /cygdrive/c/Users/kyumin/AppData/Roaming/SPB_Data/m
f3
[New Thread 22164.0x5d7c]
[New Thread 22164.0xe7c]
[New Thread 22164.0x105c]

Thread 1 "mycat3" hit Breakpoint 1, main (argc=4, argv=0xa00001a70)
  at mycat3.c:12
12      show_file(argv[1]);
(gdb) p argc
$1 = 4
(gdb) p argv[1]
$2 = 0x7ffffcc84 "f1"
(gdb) p argv[2]
$3 = 0x7ffffcc87 "f2"
(gdb) p argv[3]
$4 = 0x7ffffcc8a "f3"
(gdb)
```

Set args f1 f2 f3 라고 입력하여 argument를 지정해줬다. 정상적으로 변수에 입력이 되었는지 확인을 위해 프로그램을 실행 후 p 명령어로 확인을 해보면 정상적으로 입력이된 것을 볼 수 있다.

```
Thread 1 "mycat3" hit Breakpoint 1, main (argc=4, argv=0xa00001a70)
  at mycat3.c:12
12      show_file(argv[1]);
(gdb) s
show_file (fname=0x7ffffcc84 "f1") at mycat3.c:16
16      x=open(fname, O_RDONLY, 00777); // open the specified file
(gdb) s
17      if (x== -1){                      // if there is an error
(gdb) s
21      display_content(x);
(gdb) s
display_content (x=3) at mycat3.c:28
28      y=read(x, buf, 20);                // read max 20 bytes
(gdb) s
29      if (y==0) break;                  // if end-of-file, get out
(gdb) s
30      write(1, buf, y);                 // write to terminal
(gdb) |
```

디버깅 모드에서 s 명령어는 다음 코드를 한 줄 씩 진행하라는 의미다. 위 사진을 보면 한 줄씩 코드가 진행되며, 함수가 나온다면 함수 내부의 코드를 한줄 씩 진행하는 것을 볼 수 있다.

```

(gdb) p argv[3]
$4 = 0x7ffffcc8a "f3"
(gdb) n
[New Thread 22164.0x1d38]
[New Thread 22164.0x485c]
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.
13      }
(gdb) n
0x00007ff88c688088 in cygwin_dll_init () from /usr/bin/cygwin1.dll
(gdb) n
Single stepping until exit from function cygwin_dll_init,
which has no line number information.
[Thread 22164.0x1d38 exited with code 0]
[Thread 22164.0xe7c exited with code 0]
[Thread 22164.0xc50 exited with code 0]
[Thread 22164.0x485c exited with code 0]
[Thread 22164.0x5d7c exited with code 0]
[Inferior 1 (process 22164) exited normally]
(gdb)

```

디버깅 모드에서 n은 s와 동일하게 한 줄씩 코드를 진행하라는 의미다. 그러나 함수가 있다면 함수 내부의 코드는 한 번에 진행한다. 위 사진을 보면 함수 내부에 들어가서 한 줄씩 진행하지 않는다. 한 번에 끝났기 때문에 프로그램이 종료된 것을 볼 수 있다.

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
    }
}

```

```

kyumin@DESKTOP-NUDFAPK ~
$ ex1 f1
Breakpoint 1 at 0x10070107: file ex1.c, line 11.
(gdb) set args f1
(gdb) r
Starting program: /cygdrive/c/Users/kyumin/AppData/Roaming/SPB_Data/ex1 f1
[New Thread 11396.0x425c]
[New Thread 11396.0x2e8c]
[New Thread 11396.0x5324]

Thread 1 "ex1" hit Breakpoint 1, main (argc=2, argv=0xa00001a70) at ex1.c:11
11      x=open(argv[1], O_RDONLY, 00777); // open the specified file
(gdb) s
12      if (x== -1){                      // if there is an error
(gdb) s
17          y=read(x, buf, 20);          // read max 20 bytes
(gdb) s
18          if (y==0) break;             // if end-of-file, get out
(gdb) p x
$1 = 3

```

F1 파일을 오픈할 때까지는 오류 없이 정상적으로 파일이 열린 것을 볼 수 있다.

```

(gdb) s
19          write(1, buf, y);             // write to terminal
(gdb) s
17          y=read(x, buf, 20);          // read max 20 bytes
(gdb) s
18          if (y==0) break;             // if end-of-file, get out
(gdb) p y
$2 = 20
(gdb) s
[New Thread 11396.0x6210]
19          write(1, buf, y);             // write to terminal
(gdb) s
17          y=read(x, buf, 20);          // read max 20 bytes
(gdb) p y
$3 = 0
(gdb) s

```

While 반복문에서 빠져나가지 못하고 무한 반복되는데, 그 이유는 if문안에 조건이 `y==0`이기 때문이다. 이 문제를 해결하기 위해서는 `y==0`으로 수정해야한다.

```

    for(;;){
        y=read(x, buf, 20);
        if (y==0) break;
        write(1, buf, y);
    }
}

```

```

kyumin@DESKTOP-NUDFAPK ~
$ ex1 f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.

```

조건을 수정한 후 정상적인 결과가 나온다.

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".

```
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

void show_file(char *filename);
void display_content(int x);

int main(int argc, char *argv[]) {
    show_file(argv[1]);
    show_file(argv[2]);
}

void show_file(char *filename) {
    int x;
    x=open(filename, O_RDONLY, 00777);
    if (x== -1) {
        perror("error in open");
        exit(1);
    }
    display_content(x);
}

void display_content(int x) {
    // display the content of file x in the
    char buf[20];
    int y;
    for(;;) {
        y=read(x, buf, 20);
        if (y==0) break;
        write(1, buf, y);
    }
}
```

9번 문제의 코드를 활용하였다. 파일 오픈과 파일 내용 출력을 함수를 따로 만들었다. 그리고 main 함수에서는 입력된 argument를 함수에 입력했다. 두 개의 파일 출력하면 되니 입력받은 두 개의 파일을 show\_file 함수에 직접 사용했다.



```

kyumin@DESKTOP-NUDFAPK ~
$ cat f1 f2
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.
hi

kyumin@DESKTOP-NUDFAPK ~
$ mycat f1 f2
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.
hi

```

f1과 f2 파일의 내용이 모두 출력된 것을 볼 수 있다. Cat과 동일한 결과가 나온다.

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.

```

#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

void show_file(char *filename);
void display_content(int x);

int main(int argc, char *argv[]){
    for(int i=1; i < argc; i++){
        show_file(argv[i]);
    }
}

void show_file(char *filename){
    int x;
    x=open(filename, O_RDONLY, 00777); //
    if (x==-1){
        perror("error in open");
        exit(1);
    }
    display_content(x);
}

void display_content(int x){
    // display the content of file x in the s
    char buf[20];
    int y;
    for(;;){
        y=read(x, buf, 20);
        if (y==0) break;
        write(1, buf, y);
    }
}

```

이전 문제에서는 main함수에서 show\_file함수를 두 번 사용해서 두 개의 파일 내용을 출력했다. 이번 문제에서는 for문을 만들어 입력받은 argument를 show\_file 함수에 순차적으로 입력되도록 만들었다. 모든 argument 개수보다 하나 적은 즉 입력된 파일들의 개수만큼만 for문이 반복되도록 만들었다. 그래서 여러개의 파일명을 입력해도 모든 파일들의 내용을 출력할 수 있는 것이다.

```
kyumin@DESKTOP-NUDFAPK ~  
$ cat f1 f2 f8  
I have a dream  
that one day  
this nation will rise up,  
live out the true meaning of its creed.  
hi  
ab  
cd  
kyumin@DESKTOP-NUDFAPK ~  
$ mycat f1 f2 f8  
I have a dream  
that one day  
this nation will rise up,  
live out the true meaning of its creed.  
hi  
ab  
cd
```

여러 개의 파일 내용이 출력된 것을 볼 수 있다.

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

\$ mycp f1 f2

will copy f1 into f2

```
#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 x1, x2;
    x1=open(argv[1], O_RDONLY, 00777); // open the file
    x2 = open(argv[2], O_RDWR | O_CREAT, 00777);
    if (x1==-1){ // if the file can't be opened
        perror("error in open"); // print error message
    }
    char buf[20];
    int y;
    for(;;){
        y=read(x1, buf, 20); // read data from x1
        if (y==0) break; // if the file is empty, break
        write(x2, buf, y); // write data to x2
    }
}
```

Mycat.c 코드를 응용해서 만들었다. 두 번째 argument 를 이름으로하는 파일을 열고 x1 으로 저장한다. 세 번째 argument 를 이름으로하는 파일을 읽기 및 쓰기로 모드로 열고(파일이 없다면 생성) x1 에서 읽은 내용을 x2 에 작성한다.

```
kyumin@DESKTOP-NUDFAPK ~
$ mycp f1 f2

kyumin@DESKTOP-NUDFAPK ~
$ cat f1
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.

kyumin@DESKTOP-NUDFAPK ~
$ cat f2
I have a dream
that one day
this nation will rise up,
live out the true meaning of its creed.

kyumin@DESKTOP-NUDFAPK ~
$ |
```

Mycp f1 f2 라고 입력 후 f2 의 내용을 확인해보면 f1 과 동일하다는 것을 알 수 있다.

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

```
$ cat f1
```

```
abc
```

```
$ xxd f1
```

```
00000000: 6162 630a
```

```
abc.
```

```
$ myxxd f1
```

```
61 62 63 a
```

```
int main(int argc, char *argv[]){
    int x;
    x=open(argv[1], O_RDONLY, 00777); // op
    if (x == -1){
        perror("error in open");
    }
    char buf[20];
    int y;
    for(;;){
        y=read(x, buf, 20);
        if (y==0) break;
        for(int i = 0; i < y; i++){
            printf("%x ", buf[i]);
        }
    }
}
```

파일을 읽고 결과를 x에 저장한다. 이후 x의 파일 내용을 읽고, buf에 저장한다. 그리고 버퍼의 배열에서 문자 하나씩 출력한다. Printf 사용 시 %x를 사용하여 16진수로 출력한다.

```
kyumin@DESKTOP-NUDFAPK ~
$ xxd f2
00000000: 6162 630a abc.

kyumin@DESKTOP-NUDFAPK ~
$ myxxd f2
61 62 63 a
```

Xxd와 동일하게 myxxd가 작동하는 것을 볼 수 있다.

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 .....
}
```

```
.....
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <dirent.h>

void optionO(char *argvcp, char *argvpa);
void optionX(char *argv);
void optionP(char *argv);
void nameprint(char *word, int count);
void optionD(char *argv);
void show_file(char *filename);
void display_content(int x);

int main(int argc, char *argv[]){
    if(strcmp(argv[1], "-o") == 0){
        optionO(argv[2], argv[3]);
    }

    else if(strcmp(argv[1], "-x") == 0){
        optionX(argv[2]);
    }
    else if(strcmp(argv[1], "-p") == 0){
        optionP(argv[2]);
    }
    else if(strcmp(argv[1], "-d") == 0){
        optionD(argv[2]);
    }

    else{
        for(int i = 1; i < argc; i++){
            show_file(argv[i]);
        }
    }
}
```

옵션별로 함수를 만들었다.

Main 함수 내부에서는 strcmp 함수를 사용해서 입력 받은 옵션이 어떤 문자인지 확인하고 옵션 함수를 실행해줬다. 만약 옵션이 없다면 입력받은 argument의 파일 내용만 출력한다.

```

void option0(char *argvcp, char *argvpa){
    int x1, x2, y;
    char buf[20];
    x1=open(argvcp, O_RDONLY, 00777); // open
    x2=open(argvpa, O_RDWR | O_CREAT, 00777);
    if (x1==-1){
        perror("error in open"); // re
        exit(1); // ar
    }
    for(;;){
        y=read(x1, buf, 20);
        if(y==0) break;
        write(x2, buf, y);
    }
}

```

-o 옵션인 경우다. 파일을 복사하는 코드이다. 세 번째로 입력된 argument가 복사할 원본 파일이고, 네 번째로 입력된 argument가 복사한 내용을 입력할 파일이다. 원본 파일의 내용을 읽고, write 함수를 사용해서 내용을 입력해줬다.

```

void optionX(char *argv){
    int x;
    x=open(argv, O_RDONLY, 00777); // open
    if (x == -1){
        perror("error in open");
    }
    char buf[20];
    int y;
    for(;;){
        y=read(x, buf, 20);
        if (y==0) break;
        for(int i = 0; i < y; i++){
            printf("%x ", buf[i]);
        }
    }
}

```

-x 옵션인 경우다. 파일의 내용을 16진수로 표기하는 코드이다. 파일의 내용을 읽고, printf를 통해 출력을 하는데 한 글자씩 16진수(%x)로 출력한다.

```

void optionP(char *argv){
    int count=0; //현재 출력할 name을 숫자로 표기
    char *y;
    char buf[20];
    char *word; //단위로 출력한 토큰
    FILE *fp = fopen(argv, "r");
    bool ln = false, exce = false; //개행문자, : 문자
    printf("id: ");
    for(;;){
        y = fgets(buf, 20, fp);

        if(y == NULL) break;

        //파일을 읽을 때 :까지 읽으면 name이 정확히
        if(buf[strlen(buf)-1] == ':') exce = true;

        word = strtok(buf, ":");
        printf("%s", word);
        count++;

        for(;;){
            //한 행의 마지막 단어라면 ln을 참으로
            //count를 0으로 초기화하기 위함임.
            if(word[strlen(word)-1] == '\n'){
                ln = true;
                printf("id: ");
            }
            word = strtok(NULL, ":");
            if(word == NULL) {
                count--; //count 줄이기
                break;
            }
            nameprint(word, count);
            count++;
        }

        //ln은 행의 마지막 단어인지 확인을 위해 만들
        //마지막 단어라면 count는 0으로 초기화해야 함
        if(ln == true){
            count = 0;
            ln = false;
        }

        //마지막 문자가 :이라면 count가 올라가지 않음
        //않으므로 따로 코드를 추가함.
        if(exce == true){
            count++;
            nameprint("", count);
            exce = false;
        }
    }
    fclose(fp);
}

```

난다면 if(ln == true)가 실행된다. 여기서는 count를 0으로 초기화하고, ln을 다시 false로 초기화한다.

만약에 원본 파일을 읽고 buf에 저장했는데 마지막 문자가 ‘:’ 이라면 count가 올라가지 않는 문제가 발생한다. 그런 경우 exce를 true로 바꾸고, 마지막에 count를 직접 올려준다.

이전에 word를 출력해준다고 했는데, 출력 시 name도 같이 출력을 해줘야한다. 그 부분은 아래에 함수를 만들어두었다.

-p 옵션인 경우다. /etc/passwd의 내용을 보기 쉽게 바꾸는 코드다.

가장 쉬운 방법은 buf를 충분히 큰 배열로 만들어주고, fgets를 통해 한 줄씩 읽고, ‘:’ 문자마다 출력을 하는데 그 앞에 이름을 출력해주면 된다. 그러나 이 경우 buf에 필요하지 않게 큰 배열을 준다면 메모리를 많이 소모하고, 만약 한 줄의 길이가 매우 길면 오류가 생길 수 있기 때문에 다른 방식으로 만들었다.

Buf의 크기를 20으로 만들었다. 우선 가장 처음에 “id: “를 출력해준다. 이후 파일을 열고, 문자를 읽어서 buf에 저장한다. 이 때 fgets를 사용했는데 이는 줄바꿈이 나타난다면 읽기를 종료하는 특성이 있다.

이후 strtok를 이용해서 ‘:’ 문자가 나타날 때까지의 스트링을 포인터 word에 지정해주고, 출력한다. 그리고 count라는 int 변수를 1 올려준다. 그 이유는 토큰을 출력할 때 그 앞에 어떤 이름인지 출력을 해줘야하니 현재 어떤 단어인지 구별을 해주기 위함이다. Buf에 저장된 스트링을 마지막까지 ‘:’ 을 찾아서 출력하고, count를 올려준다. 여기서 마지막 단어를 출력할 때는 ‘:’ 을 만나지 않아도 count가 올라간다. 그래서 마지막에 if(word == NULL)에서 count를 1 낮춰주는 코드가 필요하다.

만약 줄바꿈이 일어난다면 처음부터, 즉 id라는 메시지 출력부터 다시 해야하므로, ln을 참으로 바꿔서 행의 마지막 부분이라는 것을 표기해둔다.

그렇게 원본 파일의 1행 출력이 모두 끝



```

void nameprint(char *word, int count){
    switch(count){
        case 0:
            printf("id: %s", word);
            break;
        case 1:
            printf(" passwd: %s", word);
            break;
        case 2:
            printf(" uid: %s", word);
            break;
        case 3:
            printf(" gid: %s", word);
            break;
        case 4:
            printf(" desc: %s", word);
            break;
        case 5:
            printf(" home: %s", word);
            break;
        case 6:
            printf(" sh: %s", word);
            break;
        default:
            printf("error");
            break;
    }
}

```

-p 옵션에서 출력을 할 때 name을 같이 출력하기 위해 사진과 같이 코드를 만들었다. optionP 함수에서 count를 저장해줬던 이유는 현재 어떤 단어가 출력되고 있는지 구별하기 위함이다. 그래서 switch를 이용해서 count에 따라 어떤 메시지가 출력 되어야하는지 따로 만들어주었다.

```

void optionD(char *argv){
    DIR *dir_ptr = NULL;
    struct dirent *file = NULL;
    char home[1024];

    strncpy(home, getenv("HOME"), sizeof(home));
    strncat(home, "/", sizeof(home)-strlen(home)-1);
    strncat(home, argv, sizeof(home)-strlen(home)-1);

    if((dir_ptr = opendir(home)) == NULL)
    {
        fprintf(stderr, "%s directory 정보를 읽을 수 없습니다.\n", home);
        exit(1);
    }

    while((file = readdir(dir_ptr)) != NULL)
    {
        printf("%s\n", file->d_name);
    }

    closedir(dir_ptr);
}

```

-d 옵션인 경우다. 입력된 디렉토리의 내부 파일을 출력하는 함수다. home 이라는 캐릭터 배열에는 환경변수 "HOME"의 위치를 저장해줬다. 그리고 입력된 argument는 strncat 함수를 이용해서 home 배열에 추가 저장해줬다. 그래서 home 배열의 주소로 파일이름(d\_name)을 출력해줬다.

```

void show_file(char *filename){
    int x;
    x=open(filename, O_RDONLY, 00777);
    if (x==-1){
        perror("error in open");
        exit(1);
    }
    display_content(x);
}

void display_content(int x){
    // display the content of file x in the terminal
    char buf[20];
    int y;
    for(;;){
        y=read(x, buf, 20);
        if (y==0) break;
        write(1, buf, y);
    }
}

```

옵션이 없는 경우다. 입력된 argument의 파일 내용을 화면에 출력하도록 만들었다.

```

kyumin@DESKTOP-NUDFAPK ~
$ mycat -o f2 f3

kyumin@DESKTOP-NUDFAPK ~
$ cat f2 f3
abc
abc

```

-o 옵션의 경우 파일이 정상적으로 복사된 것을 볼 수 있다.

```

kyumin@DESKTOP-NUDFAPK ~
$ mycat -x f1
49 20 68 61 76 65 20 61 20 64 72 65 61 6d a 74 68 61 74 20 6f 6e 65 20 64 61 79 a
74 68 69 73 20 6e 61 74 69 6f 6e 20 77 69 6c 6c 20 72 69 73 65 20 75 70 2c a 6c 69
76 65 20 6f 75 74 20 74 68 65 20 74 72 75 65 20 6d 65 61 6e 69 6e 67 20 6f 66 20
69 74 73 20 63 72 65 65 64 2e a
kyumin@DESKTOP-NUDFAPK ~
$ xxd f1
00000000: 4920 6861 7665 2061 2064 7265 616d 0a74 I have a dream.t
00000010: 6861 7420 6f6e 6520 6461 790a 7468 6973 hat one day.this
00000020: 206e 6174 696f 6e20 7769 6c6c 2072 6973 nation will ris
00000030: 6520 7570 2c0a 6c69 7665 206f 7574 2074 e up,.live out t
00000040: 6865 2074 7275 6520 6d65 616e 696e 6720 he true meaning
00000050: 6f66 2069 7473 2063 7265 6564 2e0a of its creed..

```

-x 옵션의 경우 16진수로 파일 내용을 출력하는 옵션이다. Xxd 명령어를 통해 확인한 내용과 동일한 것을 볼 수 있다.

```

kyumin@DESKTOP-NUDFAPK ~
$ mycat -p /etc/passwd
id: SYSTEM passwd: * uid: 18 gid: 18 desc: U-NT AUTHORITY\SYSTEM,S-1-5-18 home: /h
ome/SYSTEM sh: /bin/bash
id: LOCAL SERVICE passwd: * uid: 19 gid: 19 desc: U-NT AUTHORITY\LOCAL SERVICE,S-1
-5-19 home: / sh: /sbin/nologin
id: NETWORK SERVICE passwd: * uid: 20 gid: 20 desc: U-NT AUTHORITY\NETWORK SERVICE
S,S-1-5-20 home: / sh: /sbin/nologin
id: Administrators passwd: * uid: 544 gid: 544 desc: U-BUILTIN\Administrators,S-1-
5-32-544 home: / sh: /sbin/nologin
id: NT SERVICE+TrustedInstaller passwd: * uid: 328384 gid: 328384 desc: U-NT SERVI
CE\TrustedInstaller,S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464
home: / sh: /sbin/nologin
id: Administrator passwd: * uid: 197108 gid: 197121 desc: U-DESKTOP-NUDFAPK\Admini
strator,S-1-5-21-3982802315-3244896599-3593029022-500 home: /home/Administrator sh
: /bin/bash
id: DefaultAccount passwd: * uid: 197111 gid: 197121 desc: U-DESKTOP-NUDFAPK\Defau
ltAccount,S-1-5-21-3982802315-3244896599-3593029022-503 home: /home/DefaultAccount
sh: /bin/bash
id: Guest passwd: * uid: 197109 gid: 197121 desc: U-DESKTOP-NUDFAPK\Guest,S-1-5-21-
3982802315-3244896599-3593029022-501 home: /home/Guest sh: /bin/bash
id: kyumin passwd: * uid: 197609 gid: 197121 desc: U-DESKTOP-NUDFAPK\kyumin,S-1-5-
21-3982802315-3244896599-3593029022-1001 home: /home/kyumin sh: /bin/bash
id: WDAGUtilityAccount passwd: * uid: 197112 gid: 197121 desc: U-DESKTOP-NUDFAPK\W
DAGUtilityAccount,S-1-5-21-3982802315-3244896599-3593029022-504 home: /home/WDAGUt
ilityAccount sh: /bin/bash
id:

```

-p 옵션을 사용한 결과다. buf를 충분히 큰 배열로 만드는 방식으로 진행했다면 더 쉽게 코드를 만들고, 필요하지 않은 메시지가 출력되지 않았을 것이다.

그러나 메모리 낭비가 심할 수 있으므로 buf를 20으로 만들어서 코드를 만들었다. 저장된 파일의 id, passwd, uid 등 모든 정보가 올바르게 출력된 것을 볼 수 있다. 이 코드는 buf가 20으로 작다는 장점이 있지만 마지막에 “id: “라는 메시지가 출력된다는 단점이 있다.

```

kyumin@DESKTOP-NUDFAPK ~
$ mycat -d d1
.
..
f3

kyumin@DESKTOP-NUDFAPK ~
$ cd d1

kyumin@DESKTOP-NUDFAPK ~/d1
$ ls -la
.  ..  f3

```

-d 옵션을 사용한 예시다. Mycat 함수를 사용해서 d1 내부의 파일을 출력했다. D1 디렉토리로 진입한 후 ls를 이용해 확인해보면 mycat과 동일한 결과를 볼 수 있다.