

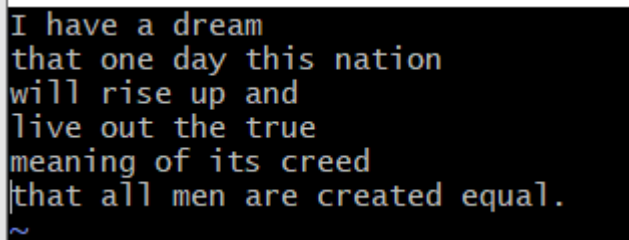
Week4 HW

12201922 이규민

1) Make a file, "f1", and fill it with more than 20 bytes.

```
$vi f1
```

```
I have a dream  
that one day this nation  
will rise up and  
live out the true  
meaning of its creed  
that all men are created equal.
```

A screenshot of a terminal window with a black background and light blue text. The text is the same as the previous block, but it is enclosed in a rectangular box that represents a terminal window. The text is: "I have a dream", "that one day this nation", "will rise up and", "live out the true", "meaning of its creed", "that all men are created equal.", followed by a tilde symbol "~" on the next line.

```
I have a dream  
that one day this nation  
will rise up and  
live out the true  
meaning of its creed  
that all men are created equal.  
~
```

Vi f1으로 코맨드 모드에 진입하여 내용을 입력하였다.

2) Try the code in 6-0), 6-1), 6-2), 6-3), 6-4), 6-5). For 6-3) explain the strange output.

```
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1  
I have a dream  
that
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1  
I have a dream  
that  
kyumin@DESKTOP-NUDFAPK ~  
$
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1  
I have a dream  
that one day this nation  
will rise up and  
live out the true  
meaning of its creed  
that all men are created equal.
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1  
I have a dream  
that one day this nation  
will rise up and  
live out the true  
meaning of its creed  
that all men are created equal.  
are create  
kyumin@DESKTOP-NUDFAPK ~  
$ |
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1  
  
kyumin@DESKTOP-NUDFAPK ~  
$
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1
```

6-3) 코드를 실행하면 마지막에 “are create”라는 문자가 추가로 붙는다. 그 이유는 `write(1, buf, 20);`에서 `y`가 아닌 `20`이라고 작성했기 때문이다.

Buf 배열은 저장되어있는 스트링보다 저장할 스트링의 길이가 짧다면 이전 문자열이 기록에 남는다. 즉 기존에 “that all men are create”라고 저장되어있다가 “d equal.\0 all ceate”라고 저장된다면 buf를 출력할 때 끝에 “are create”가 추가로 출력되는 것이다.

3) Find the byte size of f2 with “ls -l f2” . Use xxd to find out the actual data stored in f2.

```
kyumin@DESKTOP-NUDFAPK ~
$ ls -l f2
-rwxr-xr-x+ 1 kyumin 없음 129 Mar 25 09:27 f2

kyumin@DESKTOP-NUDFAPK ~
$ xxd f2
00000000: 4920 6861 7665 2061 2064 7265 616d 0a74 I have a dream.t
00000010: 6861 7420 6f6e 6520 6461 7920 7468 6973 hat one day this
00000020: 206e 6174 696f 6e0a 7769 6c6c 2072 6973 nation.will ris
00000030: 6520 7570 2061 6e64 0a6c 6976 6520 6f75 e up and.live ou
00000040: 7420 7468 6520 7472 7565 200a 6d65 616e t the true .mean
00000050: 696e 6720 6f66 2069 7473 2063 7265 6564 ing of its creed
00000060: 0a74 6861 7420 616c 6c20 6d65 6e20 6172 .that all men ar
00000070: 6520 6372 6561 7465 6420 6571 7561 6c2e e created equal.
00000080: 0a
.
```

Ls -l f2명령어를 사용하여 확인한 크기는 129 바이트다. Xxd를 사용하여서도 확인해봤다. 16진수로 하나에 1바이트다. 예를들어 4920 에서는 49와 20 이렇게 2바이트다. 총 개수를 세어보면 129 바이트로 확인된다.

3-1) Write a program that counts the number of bytes in f2. Compare it with the output of “ls -l f2” .

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

int main(){
    int x, y,R=0;
    char buf[20];

    x=open("f2", O_RDONLY, 00777);
    for(;;){
        y=R;
        R+=read(x,buf,20);
        if(y==R) break;
    }
    printf("The number of bytes is %d\n",y);
    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o ex1 ex1.c
kyumin@DESKTOP-NUDFAPK ~
$ ./ex1
The number of bytes is 129
```

우선 읽은 문자열을 저장할 buf 배열을 만들었다. f2파일을 읽기모드로 열었다. 그리고 x 파일을 읽은 내용을 buf에 저장하고, 읽은 길이를 변수 R에 더해준다. 만약 읽은 글자 수가 200 이하라면 읽은만큼만 y에 더해줄 수 있다. 이렇게 무한 for문을 돌려주는데 위 코드처럼 y=R; / if(y==R) break; 명령어를 사용해줬는데 이는 R의 값에 변화가 없다면 무한 for문을 멈추라는 의미다. Y 값을 출력해보면 f1파일이 몇 바이트인지 알 수 있다.

4) Write a program "hw4.c" that opens f2 and shows each byte of it in hexadecimal number, decimal number, and character. Use printf("%x %d %c\\n",) to display a number in various format.

```
int x, y; char buf[20];

x=open("f2", O_RDONLY, 00777);    // open f2 for reading

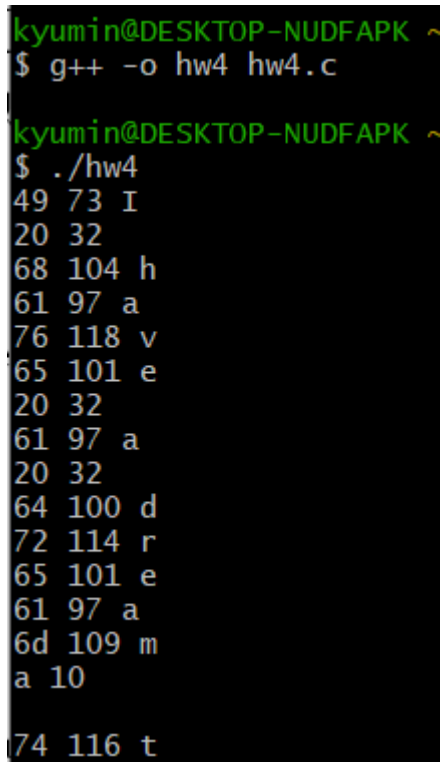
for(;;){

    y=read(x, buf, 1); // read next byte

    if (y==0) break;    // we read all, exit the loop

    printf("%x %d %c\\n", buf[0], buf[0], buf[0]); // display

}
```



```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o hw4 hw4.c

kyumin@DESKTOP-NUDFAPK ~
$ ./hw4
49 73 I
20 32
68 104 h
61 97 a
76 118 v
65 101 e
20 32
61 97 a
20 32
64 100 d
72 114 r
65 101 e
61 97 a
6d 109 m
a 10

74 116 t
```

printf("%x %d %c\\n",buf[0]...)을 사용하여 16진수 / 10진수 / 문자 형태로 출력이 가능하다.

5) Compile hw4.c with -g option and run gdb to execute each instruction one by one. Use “p” or “x” to check the value of a variable. For ml mac, use lldb instead of gdb.

```
$ gcc -g -o hw4 hw4.c
$ gdb hw4
gdb) b main -- stop at main
gdb) r -- run
.....
9 x=open("f2", O_RDONLY, 00777); -- next line to execute
gdb) list -- show code list
gdb) n -- execute current line
11 y=read(x, buf, 1); -- line 9 has been executed. next is line 11
gdb) p x -- show x
$1 = 7 -- f2 is now file number 7
gdb) n
.....
gdb) p y
$2 = 1 -- we have read 1 byte
gdb) p buf[0]
$4 = 73 'I' -- assume we have 'I' in buf[0]
gdb) x/4xb buf -- show 4 bytes at buf in hexadecimal num
0x7fffffff470: 0x49 0x06 0x40 0x00 -- we have 0x49=73='I' in buf[0]
(for lldb, you need an address to use x command.
p &buf -- print the address of buf
0x0016fdff496
x/4xb 0x0016fdff496 -- show 4 bytes at addr 0x0016fdff496
)
gdb) n
..... -- repeat a few times
gdb) list -- show rest of code
gdb) b 15 -- break point at line 15 (after loop)
gdb) c -- continue to that break point
```

`gdb) q`

`-- stop gdb`

```
kyumin@DESKTOP-NUDFAPK ~
$ gdb hw4
GNU gdb (GDB) (Cygwin 13.2-1) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-cygwin".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hw4...
(gdb) b main
Breakpoint 1 at 0x10040108d: file hw4.c, line 11.
```

Gdb 명령어는 디버깅 시 사용된다. 위 사진처럼 `b main`이라고 입력하면 main에 브레이크 포인트를 걸 수 있다.

```
(gdb) r
Starting program: /cygdrive/c/Users/kyumin/AppData/Roaming/SPB_Data/hw4
[New Thread 10592.0x606c]
[New Thread 10592.0x552c]
[New Thread 10592.0xa48]

Thread 1 "hw4" hit Breakpoint 1, main () at hw4.c:11
11      x=open("f2", O_RDONLY, 00777);      // open f2 for reading
(gdb) list
6      #include <string.h>
7
8      int main(){
9          int x, y;
10         char buf[20];
11         x=open("f2", O_RDONLY, 00777);      // open f2 for reading
12         for(;;){
13             y=read(x, buf, 1); // read next byte
14             if (y==0) break;    // we read all, exit the loop
15             printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display
(gdb)
```

R을 입력하여 프로그램을 실행했고, 이전에 브레이크 걸었던 main에서 멈춘 것을 알 수 있다.

List를 입력하면 해당 프로그램의 코드를 현재 위치부터 10줄까지 볼 수 있다.

```

(gdb) n
[New Thread 10592.0x5268]
[New Thread 10592.0x6ef0]
13                 y=read(x, buf, 1); // read next byte
(gdb) p x
$1 = 3
(gdb) n
14                 if (y==0) break; // we read all, exit the loop
(gdb) p y
$2 = 1
(gdb) p buf[0]
$3 = 73 'I'
(gdb) x/4xb buf
0x7ffffcbe0: 0x49 0xcd 0xff 0xff
(gdb) n
15                 printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display

```

N은 다음 코드를 실행하는 명령어다. P x는 x의 변수를 보여주는 명령어다. x/4xb는 Buf 배열에서 4바이트 즉 4글자를 16진수로 보여주는 명령어다.

다시 n을 입력하여 다음 코드를 실행하였다.

```

(gdb) list
10                 char buf[20];
11                 x=open("f2", O_RDONLY, 00777); // open f2 for reading
12                 for(;;){
13                     y=read(x, buf, 1); // read next byte
14                     if (y==0) break; // we read all, exit the loop
15                     printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display
16                 }
17             }
(gdb) b 15
Breakpoint 2 at 0x1004010c7: file hw4.c, line 15.
(gdb) c
Continuing.
49 73 I

Thread 1 "hw4" hit Breakpoint 2, main () at hw4.c:15
15                 printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display
(gdb) q
A debugging session is active.

        Inferior 1 [process 10592] will be killed.

Quit anyway? (y or n) y

```

List는 현위치부터 10줄까지의 코드를 보여주는 명령어지만 남은 코드가 10줄이 되지 않으므로 8줄만 출력된 것을 볼 수 있다.

B 15는 15행에 브레이크 포인트를 거는 명령어다.

C는 다음 브레이크 포인트까지 실행하는 명령어다.

Q는 프로그램을 종료하는 명령어고 y를 한 번 더 눌러 종료했다.

6) Write a program that creates a file and writes "how are you doing?" in it. Use open() and write(). Confirm the result with "cat".

```
x = open("f3", O_RDWR | O_CREAT | O_TRUNC, 00777); // create f3  
  
write(x, "how are you doing?", 18); // write 18 bytes in f3
```

```
#include <fcntl.h>  
#include <sys/stat.h>  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
#include <string.h>  
  
int main(){  
    int x;  
    x = open("f3", O_RDWR | O_CREAT | O_TRUNC, 00777); // create f3  
    write(x, "how are you doing?", 18); // write 18 bytes in f3  
  
    return 0;  
}
```

X = open("f3" ,O_RDWR | O_CREAT은 f3 파일을 읽기 및 쓰기 모드로 열고, 만약 파일이 존재하지 않는다면 파일을 생성하라는 의미다. O_TRUNC는 파일에 이미 데이터가 존재한다면 데이터를 제거하라는 의미다.

Write는 x 파일에 문자열을 18 바이트만큼 입력하라는 의미다. 위에서 f3 파일을 열 때 읽기 및 쓰기 모드로 열었기 때문에 입력이 가능하다.

```
kyumin@DESKTOP-NUDFAPK ~  
$ g++ -o ex1 ex1.c  
  
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex1
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ cat f3  
how are you doing?
```

프로그램을 실행한 후 f3 파일의 내용을 확인해보면 정상적으로 입력이 된 것을 볼 수 있다.

6-1) Repeat Problem 6 but pass a string variable to "write" this time.

```
x = open("f3", O_RDWR | O_CREAT | O_TRUNC, 00777); // create f3  
  
.....  
  
write(x,y, strlen(y)); // y is a string variable that has "how ..." string
```



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

int main(){
    int x;
    char y[19];
    strcpy(y,"how are you doing?");

    x = open("f3", O_RDWR | O_CREAT | O_TRUNC, 00777);
    write(x, y, strlen(y));

    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o ex1 ex1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./ex1

kyumin@DESKTOP-NUDFAPK ~
$ cat f3
how are you doing?
kyumin@DESKTOP-NUDFAPK ~
$ echo hello > f3

kyumin@DESKTOP-NUDFAPK ~
$ cat f3
hello

kyumin@DESKTOP-NUDFAPK ~
$ ./ex1

kyumin@DESKTOP-NUDFAPK ~
$ cat f3
how are you doing?
```

6번과 다른 점은 y라는 스트링 배열을 만들었고, y에 스트링을 입력한 후 그 내용을 x에 입력했다는 점이다. F3의 내용을 다른 내용으로 바꾼 후 다시 ex1프로그램을 실행해보면 f3의 내용을 “how are you doing?” 으로 바뀐 것을 볼 수 있다.

7) Write a program that makes a copy for file "hw4.c" into another file "cphw4.c". Use open(), read(), and write(). Confirm that they are same with "cat" and "ls -l".

```
x1 = open hw4.c as O_RDONLY
x2 = open cphw4.c as O_RDWR | O_CREAT | O_TRUNC
for(;;){
    y = read max 20 bytes from x1 into buf
    if y is 0, break
    write y bytes of buf into x2
}
```

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

int main(){
    int x1,x2,y;
    char buf[20];

    x1 = open("hw4.c", O_RDONLY, 00777);
    x2 = open("cphw4.c", O_RDWR | O_CREAT | O_TRUNC, 00777);
    for(;;){
        y=read(x1,buf,20);
        if(y==0) break;
        write(x2,buf,y);        //y는 읽은 글자 수
    }

    return 0;
}
```

x1은 읽기모드로 x2는 읽기 및 쓰기 모드로 열었다. 그리고 무한 for문을 만들어서 x1의 파일 내용을 buf 배열에 저장한 후 x2 파일에 저장하도록 만들었다. 그리고 읽는 글자가 없을 때 무한 for문을 빠져나오도록 만들었다.

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o ex1 ex1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./ex1
```

```

kyumin@DESKTOP-NUDFAPK ~
$ cat cphw4.c
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(){
    int x, y;
    char buf[20];
    x=open("f2", O_RDONLY, 00777);    // open f2 for reading
    for(;;){
        y=read(x, buf, 1); // read next byte
        if (y==0) break;    // we read all, exit the loop
        printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display
    }
}

```

cat명령어를 사용해서 cphw4.c 파일의 내용을 확인해보면 Hw4.c 파일에 작성했던 코드랑 동일한 내용임을 알 수 있다.

```

kyumin@DESKTOP-NUDFAPK ~
$ ls -l
total 629
drwx-----+ 1 kyumin ㄱㄴ 0 Mar 15 2021 cdssetup
-rw-r--r--+ 1 kyumin ㄱㄴ 1945 Mar 24 00:15 char1.c
-rwxr-xr-x+ 1 kyumin ㄱㄴ 68715 Mar 23 23:55 char1.exe
-rwxr-xr-x+ 1 kyumin ㄱㄴ 430 Mar 27 10:02 cphw4.c
drwxr-xr-x+ 1 kyumin ㄱㄴ 0 Mar 9 22:13 d1
drwxr-xr-x+ 1 kyumin ㄱㄴ 0 Mar 6 12:47 d2
-rw-r--r--+ 1 kyumin ㄱㄴ 376 Mar 27 10:01 ex1.c
-rwxr-xr-x+ 1 kyumin ㄱㄴ 66197 Mar 27 10:01 ex1.exe
-rwxr-xr-x+ 1 kyumin ㄱㄴ 65816 Mar 14 23:01 ex2.exe
-rw-r--r--+ 1 kyumin ㄱㄴ 221 Mar 16 16:40 ex3.c
-rwxr-xr-x+ 1 kyumin ㄱㄴ 66407 Mar 16 16:41 ex3.exe
-rw-r--r--+ 1 kyumin ㄱㄴ 54 Mar 14 12:57 ex4.c
drwxr-xr-x+ 1 kyumin ㄱㄴ 0 Mar 14 13:02 exdir
-rw-r--r--+ 1 kyumin ㄱㄴ 129 Mar 25 09:18 f1
-rwxr-xr-x+ 1 kyumin ㄱㄴ 129 Mar 25 09:27 f2
-rwxr-xr-x+ 1 kyumin ㄱㄴ 18 Mar 27 09:46 f3
-rw-r--r--+ 1 kyumin ㄱㄴ 430 Mar 25 10:45 hw4.c
-rwxr-xr-x+ 1 kyumin ㄱㄴ 66926 Mar 26 14:55 hw4.exe
-rw-r--r--+ 1 kyumin ㄱㄴ 279742 Mar 14 12:51 x
-rw-r--r--+ 1 kyumin ㄱㄴ 54 Mar 14 12:57 y.c

```

Ls -l 명령어로 파일을 확인해보면 동일한 크기로 파일이 생성된 것을 볼 수 있다.

8) Write a program that makes a copy for file "hw4" (the executable file for "hw4.c) into another file cphw4. Confirm that they are same with "xxd" and "ls -l".

Execute cphw4 to see if it runs ok.

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

int main(){
    int x1,x2,y;
    char buf[20];

    x1 = open("hw4", O_RDONLY, 00777);
    x2 = open("cphw4", O_RDWR | O_CREAT | O_TRUNC, 00777);
    for(;;){
        y=read(x1,buf,20);
        if(y==0) break;
        write(x2,buf,y);        //y는 읽은 글자 수
    }

    return 0;
}
```

Hw4파일을 읽고 buf 배열에 저장한 다음 cphw4에 복사하는 코드이다.

```
kyumin@DESKTOP-NUDFAPK ~
$ xxd cphw4 | head -8
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ.....
00000010: b800 0000 0000 0000 4000 0000 0000 0000  .....@.....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000030: 0000 0000 0000 0000 0000 0000 8000 0000  .....
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  .....!..L.!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode....$.

kyumin@DESKTOP-NUDFAPK ~
$ xxd hw4 | head -8
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000  MZ.....
00000010: b800 0000 0000 0000 4000 0000 0000 0000  .....@.....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000030: 0000 0000 0000 0000 0000 0000 8000 0000  .....
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468  .....!..L.!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f  is program canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320  t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000  mode....$.

```

Xxd 명령어로 cphw4의 내용과 hw4의 내용을 16진수로 비교해봤을 때 동일한 것을 볼 수 있다.(길

이가 길어 8줄만 출력하였다.)

```
kyumin@DESKTOP-NUDFAPK ~
$ ls -l
total 697
drwx-----+ 1 kyumin ㄹ ㄹ ㄹ 0 Mar 15 2021 cdssetup
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 1945 Mar 24 00:15 char1.c
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 68715 Mar 23 23:55 char1.exe
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 66926 Mar 28 15:15 cphw4
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 430 Mar 27 10:02 cphw4.c
drwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 0 Mar 9 22:13 d1
drwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 0 Mar 6 12:47 d2
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 372 Mar 28 15:13 ex1.c
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 66197 Mar 28 15:15 ex1.exe
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 65816 Mar 14 23:01 ex2.exe
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 221 Mar 16 16:40 ex3.c
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 66407 Mar 16 16:41 ex3.exe
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 54 Mar 14 12:57 ex4.c
drwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 0 Mar 14 13:02 exdir
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 129 Mar 25 09:18 f1
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 129 Mar 25 09:27 f2
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 18 Mar 27 09:46 f3
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 430 Mar 25 10:45 hw4.c
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ ㄹ 66926 Mar 26 14:55 hw4.exe
-rw-r--r--+ 1 kyumin ㄹ ㄹ ㄹ 279742 Mar 14 12:51 x
```

두 파일의 크기가 동일한 것을 볼 수 있다.

```
kyumin@DESKTOP-NUDFAPK ~
$ ./hw4
49 73 I
20 32
68 104 h
61 97 a
76 118 v
65 101 e
20 32
61 97 a
20 32
64 100 d
72 114 r
65 101 e
61 97 a
6d 109 m
a 10

kyumin@DESKTOP-NUDFAPK ~
$ ./cphw4
49 73 I
20 32
68 104 h
61 97 a
76 118 v
65 101 e
20 32
61 97 a
20 32
64 100 d
72 114 r
65 101 e
61 97 a
6d 109 m
a 10
74 116 t
68 104 h
61 97 a
```

두 파일 모두 정상적으로 실행되고 동일한 결과가 나오는 것을 볼 수 있다.

9) Repeat 7). But get the name of the files from the user. Confirm that the result of copy with "cat" and "ls -l".

Enter src file name

hw4.c

Enter dest file name

newhw4.c

hw4.c is copied into newhw4.c successfully.

```
int main(){
    int x1,x2,y;
    char buf[20];
    char S[20]="new";

    scanf("%s",buf);
    strcat(S,buf);

    x1 = open("hw4.c", O_RDONLY, 00777);
    x2 = open(S, O_RDWR | O_CREAT | O_TRUNC, 00777);
    for(;;){
        y=read(x1,buf,20);
        if(y==0) break;
        write(x2,buf,y);        //y는 읽은 글자 수
    }
    printf("hw4.c is copied into %s successfully.\n",S);

    return 0;
}
```

캐릭터 배열 S를 만들어 “new” 라는 스트링을 저장하고, buf 배열에 파일명을 입력받았다. 그리고 strcat 명령어로 S 배열에 buf의 내용을 더했고, 그 S의 스트링을 파일 명으로 하도록 만들었다.

```
kyumin@DESKTOP-NUDFAPK ~
$ vi ex1.c

kyumin@DESKTOP-NUDFAPK ~
$ g++ -o ex1 ex1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./ex1
hw4.c
hw4.c is copied into newhw4.c successfully.
```

```

kyumin@DESKTOP-NUDFAPK ~
$ cat newhw4.c
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(){
    int x, y;
    char buf[20];
    x=open("f2", O_RDONLY, 00777);    // open f2 for reading
    for(;;){
        y=read(x, buf, 1); // read next byte
        if (y==0) break;    // we read all, exit the loop
        printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display
    }
}

```

파일을 실행한 후 hw4.c라고 입력하니 newhw4.c라는 파일이 생성되었고, 그 내용을 보면 hw4.c랑 동일한 내용임을 알 수 있다.

```

total 699
drwx-----+ 1 kyumin 없음      0 Mar 15  2021 cdssetup
-rw-r--r--+ 1 kyumin 없음    1945 Mar 24 00:15 char1.c
-rwxr-xr-x+ 1 kyumin 없음  68715 Mar 23 23:55 char1.exe
-rwxr-xr-x+ 1 kyumin 없음  66926 Mar 28 15:15 cphw4
-rwxr-xr-x+ 1 kyumin 없음   430 Mar 27 10:02 cphw4.c
drwxr-xr-x+ 1 kyumin 없음      0 Mar  9 22:13 d1
drwxr-xr-x+ 1 kyumin 없음      0 Mar  6 12:47 d2
-rw-r--r--+ 1 kyumin 없음   477 Mar 28 16:12 ex1.c
-rwxr-xr-x+ 1 kyumin 없음  67287 Mar 28 16:12 ex1.exe
-rwxr-xr-x+ 1 kyumin 없음  65816 Mar 14 23:01 ex2.exe
-rw-r--r--+ 1 kyumin 없음   221 Mar 16 16:40 ex3.c
-rwxr-xr-x+ 1 kyumin 없음  66407 Mar 16 16:41 ex3.exe
-rw-r--r--+ 1 kyumin 없음    54 Mar 14 12:57 ex4.c
drwxr-xr-x+ 1 kyumin 없음      0 Mar 14 13:02 exdir
-rw-r--r--+ 1 kyumin 없음   129 Mar 25 09:18 f1
-rwxr-xr-x+ 1 kyumin 없음   129 Mar 25 09:27 f2
-rwxr-xr-x+ 1 kyumin 없음    18 Mar 27 09:46 f3
-rw-r--r--+ 1 kyumin 없음   430 Mar 25 10:45 hw4.c
-rwxr-xr-x+ 1 kyumin 없음  66926 Mar 26 14:55 hw4.exe
-rwxr-xr-x+ 1 kyumin 없음   430 Mar 28 16:12 newhw4
-rwxr-xr-x+ 1 kyumin 없음   430 Mar 28 16:13 newhw4.c
-rw-r--r--+ 1 kyumin 없음 279742 Mar 14 12:51 x

```

Ls -l 명령어로 hw4.c와 newhw4.c를 비교해보니 파일 크기가 동일한 것을 볼 수 있다.

10) Write "mycat" that displays the contents of a user-input file in the terminal in characters. Give a text file and a non-text file to mycat and explain the difference.

```
$/mycat
```

```
Enter file name
```

```
f1
```

```
The content of f1 is :
```

```
I have a dream
```

```
that one day this nation
```

```
will rise up and
```

```
live out the true
```

```
meaning of its creed
```

```
that all men are created equal.
```

```
$/mycat
```

```
Enter file name
```

```
hw4
```

```
.....
```

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

int main(){
    int x,y;
    char buf[20];

    printf("Enter file name\n");
    scanf("%s",buf);
    printf("The content of %s is :\n",buf);
    x = open(buf, O_RDONLY | O_CREAT, 00777);
    for(;;){
        y=read(x,buf,20);
        if(y==0) break;
        write(1,buf,y);    //y는 읽은 글자 수
    }

    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ ./mycat
Enter file name
f1
The content of f1 is :
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.
```

코드는 스트링을 입력 받은 후 buf 배열에 저장하고, 파일명이 buf의 스트링인 파일을 엽니다. 그리고 그 x 파일을 20글자씩 읽어서 화면에 출력하도록 write(1,buf,y) 명령어를 사용했다. 여기서 1은 화면에 출력한다는 뜻이다.

그리고 프로그램 실행 후 f1 파일을 실행한 후 f1을 입력하면 f1의 내용들이 출력되는 것을 볼 수 있다.


```

kyumin@DESKTOP-NUDFAPK ~
$ ./mycat
Enter file name
hw4
The content of hw4 is :
MZ@L!This program cannot be run in DOS mode.
$PEd_mcf&
@)P
@E.text''.data' @.rdata0@@.buildid5@@.pdataP@@.xda
tad'@@.bssp.idata@.rsrc@@.reloc
"@B/4'$@B/19
0@B/45@1
ff.D UH H @A H aH E H U E A E } t. E E E A
A b H H M H @]D % o % o % o % o %
o % o H (1 U oo D
h H H 5j H H ]H 58H H H 5f H H 4H ^H H
H % 5 H sxH )H H H H C] H =C H *H 5(H
H H
H H
H PH
H PH

```

프로그램 실행 후 Hw4입력하면 hw4의 내용이 출력된다. 그 파일은 실행파일이고, 그 안에는 기계어가 들어가있으므로 이상한 문자들이 출력된다.

11) Write "myxxd" that displays the contents of a user-input file in the terminal in hexadecimal numbers. Give a text file and a non-text file to myxxd and explain the difference. You need to use printf("%x ", buf[i]) to display a byte in a hexadecimal number. Also declare the buffer as an array of unsigned char. Compare the result from the result of xxd.

```
$/myxxd
```

```
Enter file name
```

```
f1
```

```
The content of f1 is :
```

```
49 20 68 61 .....
```

```
$ xxd f1
```

```
.....
```

```
$/myxxd
```

```
Enter file name
```

```
hw4
```

```
.....
```

```
$ xxd hw4
```

```
.....
```

```
int main(){
    int x,y;
    unsigned char buf[20];

    printf("Enter file name\n");
    scanf("%s",buf);
    printf("The content of %s is :\n",buf);
    x = open(buf, O_RDONLY | O_CREAT, 00777);
    for(;;){
        y=read(x,buf,20);
        if(y==0) break;
        int i;
        for(i=0;i<y;i++) printf("%x ",buf[i]);

    }

    return 0;
}
```

Buf는 캐릭터 형태이므로 음수가 필요 없어 unsigned로 선언해줬다. 그리고 입력받은 내용을 buf에 저장하고, buf에 저장된 스트링의 파일명인 파일을 열었다. 그리고 그 파일의 내용을 buf에 저장하고, printf를 통해 문자를 16진수 형태로 출력했다.

```

kyumin@DESKTOP-NUDFAPK ~
$ ./myxxd
Enter file name
f1
The content of f1 is :
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
20 74 68 69 73 20 6e 61 74 69 6f 6e a 77 69 6c 6c 20 72 69 73 65 20 75 70 20 61
6e 64 a 6c 69 76 65 20 6f 75 74 20 74 68 65 20 74 72 75 65 20 a 6d 65 61 6e 69 6
e 67 20 6f 66 20 69 74 73 20 63 72 65 65 64 a 74 68 61 74 20 61 6c 6c 20 6d 65 6
e 20 61 72 65 20 63 72 65 61 74 65 64 20 65 71 75 61 6c 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 7920 7468 6973 hat one day this
00000020: 206e 6174 696f 6e0a 7769 6c6c 2072 6973 nation.will ris
00000030: 6520 7570 2061 6e64 0a6c 6976 6520 6f75 e up and.live ou
00000040: 7420 7468 6520 7472 7565 200a 6d65 616e t the true .mean
00000050: 696e 6720 6f66 2069 7473 2063 7265 6564 ing of its creed
00000060: 0a74 6861 7420 616c 6c20 6d65 6e20 6172 .that all men ar
00000070: 6520 6372 6561 7465 6420 6571 7561 6c2e e created equal.
00000080: 0a

```

프로그램을 통해 f1을 16진수형태로 확인했고, xxd를 통해 16진수 형태로 출력했다. 두 내용은 동일한 것을 볼 수 있다.

```

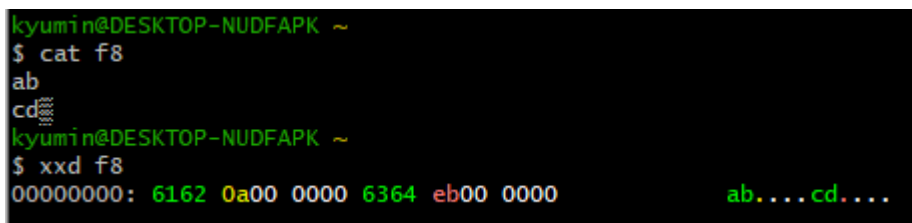
kyumin@DESKTOP-NUDFAPK ~
$ ./myxxd
Enter file name
hw4
The content of hw4 is :
4d 5a ffffffff90 0 3 0 0 0 4 0 0 0 ffffffff ffffffff 0 0 fffffffb8 0 0 0 0 0 0 40
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ffffffff80
0 0 0 e 1f fffffffba e 0 fffffffb4 9 fffffffcd 21 fffffffb8 1 4c fffffffcd 21 54 68 6
9 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e 20 69 6e
20 44 4f 53 20 6d 6f 64 65 2e d d a 24 0 0 0 0 0 0 0 50 45 0 0 64 fffffff86 12 0
6d 63 2 66 0 fffffffcc 0 0 fffffffa7 2 0 0 fffffff0 0 26 0 b 2 2 2a 0 8 0 0 0 1e 0
0 0 2 0 0 0 10 0 0 0 10 0 0 0 0 40 0 1 0 0 0 0 10 0 0 0 2 0 0 4 0 0 0 0 0 0 5
0 2 0 0 0 0 0 0 fffffffa0 1 0 0 6 0 0 ffffffffa 29 1 0 3 0 0 fffffff80 0 0 20 0 0
0 0 0 0 10 0 0 0 0 0 0 0 10 0 0 0 0 10 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0
0 0 0 0 0 0 fffffff80 0 0 fffffff9c 2 0 0 fffffff90 0 0 fffffffe8 4 0 0 0 50 0 0 f
ffffffc0 0 0 0 0 0 0 0 0 0 0 0 fffffffa0 0 0 c 0 0 0 0 40 0 0 1c 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 fffffffd0
ffffff80 0 0 fffffff90 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2e 7
kyumin@DESKTOP-NUDFAPK ~
$ xxd hw4
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000 MZ.....
00000010: b800 0000 0000 0000 4000 0000 0000 0000 .....@.....
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000030: 0000 0000 0000 0000 0000 0000 8000 0000 .....
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468 .....!..L.!Th
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f is program canno
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320 t be run in DOS
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000 mode....$.
00000080: 5045 0000 6486 1200 6d63 0266 00cc 0000 PE..d...mc.f...
00000090: a702 0000 f000 2600 0b02 022a 0008 0000 .....&...*...
000000a0: 001e 0000 0002 0000 0010 0000 0010 0000 .....
000000b0: 0000 4000 0100 0000 0010 0000 0002 0000 ..@.....
000000c0: 0400 0000 0000 0000 0500 0200 0000 0000 .....
000000d0: 00a0 0100 0006 0000 fa29 0100 0300 0080 .....).
000000e0: 0000 2000 0000 0000 0010 0000 0000 0000 .....

```

프로그램을 통해 hw4을 16진수형태로 확인했고, xxd를 통해 16진수 형태로 출력했다. 두 내용은 동일한 것을 볼 수 있다.

12) Run following code and display f8 with cat and xxd respectively. Explain the results.

```
int x;  
  
x=open("f8", O_CREAT|O_RDWR|O_TRUNC, 00777);  
  
write(x, "ab", 2);  
  
int y=10;  
  
write(x, &y, 4);  
  
write(x, "cd", 2);  
  
y=235;  
  
write(x, &y, 4);
```



```
kyumin@DESKTOP-NUDFAPK ~  
$ cat f8  
ab  
cd  
kyumin@DESKTOP-NUDFAPK ~  
$ xxd f8  
00000000: 6162 0a00 0000 6364 eb00 0000 ab....cd....
```

우선 코드에 대한 분석을 해보면 write(x, "ab", 2)는 ab 스트링을 가리키는 주소에서 2글자만큼 x에 저장하라는 의미다. 즉 x에 ab라는 문자를 저장하라는 뜻이다. write(x, &y, 4)도 y의 주소에서 4개의 글씨를 x에 저장하라는 의미다.

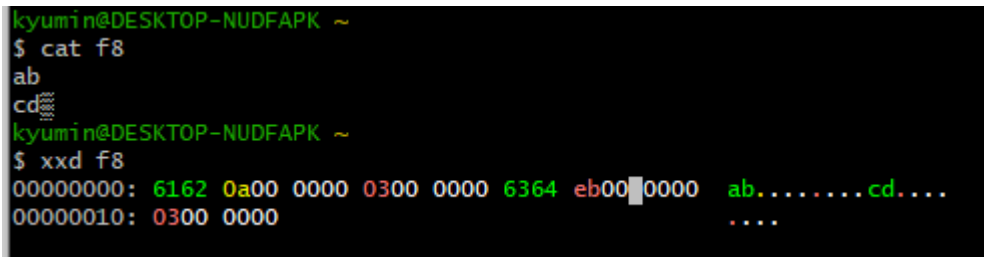
위 코드를 실행해보면 f8 파일이 생성된다. Cat 명령어로 그 파일의 내용을 확인해보면 ab가 출력되고 줄바꿈이 일어난 것을 볼 수 있다. Y의 값을 10으로 저장했었기 때문에 x에는 10 즉 개행문자(LF)가 추가되었기 때문에 줄바꿈이 일어난 것이다. 그리고 write(x, &y, 4)에서 4바이트를 저장하라고했으나 저장된 문자가 없어 내용이 입력되지는 않았다. 그리고 cd입력 후 문자는 아니지만 235라는 숫자가 저장되었다. 그래서인지 깨진 문자가 보인다.

F8 파일을 xxd로 확인해보면 0a00 0000과 eb00 0000이 보이는데 이는 y에 저장된 값은 하나인데 저장하라고 명령을 한 것은 4글자이니 00이 입력되었기 때문이다. 여기서 0a와 eb를 10진수로 환산해보면 10과 235다. 이는 코드에서 저장했던 y의 값이다.

12-1) Run following code and display f8 with cat and xxd respectively. Explain the results.

```
int x;  
  
x=open("f8", O_CREAT|O_RDWR|O_TRUNC, 00777);  
  
write(x, "ab", 2);  
  
int y=10;  
  
write(x, &y, 8);  
  
write(x, "cd", 2);
```

```
y=235;  
  
write(x, &y, 8);
```



```
kyumin@DESKTOP-NUDFAPK ~  
$ cat f8  
ab  
cd  
kyumin@DESKTOP-NUDFAPK ~  
$ xxd f8  
00000000: 6162 0a00 0000 0300 0000 6364 eb00 0000  ab.....cd....  
00000010: 0300 0000                                     ....
```

이번 코드는 `write(x, &y, 8)`을 입력했기 때문에 `xxd`로 결과화면의 우측부분에 . 표시가 8개로 늘었다. 중간에 0300에서 03 이 어떤 의미인지는 알 수 없다.

13) Write a program that divides a given file into three small files of roughly equal size. Use fstat() to find out the size of a file.

Enter file name to split

f9

f9 is split into f91, f92, and f93.

```
int main(){
    int N=3,x,x1,x2,x3,y,a,b,c;
    char buf[20];
    char *fName[20];

    fName[0]=new char[20];
    scanf("%s",fName[0]);

    x=open(fName[0],O_RDONLY | O_CREAT, 00777);

    int i=1;
    for(i=1;i<=N;i++){
        fName[i]=new char[20];
        sprintf(fName[i],"%s%d",fName[0],i);
    }

    x1=open(fName[1],O_RDWR | O_CREAT | O_TRUNC, 00777);
    x2=open(fName[2],O_RDWR | O_CREAT | O_TRUNC, 00777);
    x3=open(fName[3],O_RDWR | O_CREAT | O_TRUNC, 00777);

    struct stat tmp_stat;
    fstat(x, &tmp_stat);
    a=(int)tmp_stat.st_size/3;
    b=(int)tmp_stat.st_size/3 + a;
    c=(int)tmp_stat.st_size;

    for(i=0;;i++){
        y=read(x,buf,1);
        if(y==0) break;
        if(i<a) write(x1,buf,y);
        else if(a<=i && i<b) write(x2,buf,y);
        else if(b<=i && i<c) write(x3,buf,y);
        else printf("ERROR");
    }

    printf("%s is split into %s, %s, and %s.",fName[0],fName[1],fName[2],fName[3]);
    return 0;
}
```

하나의 파일을 비슷한 사이즈인 세 개의 파일로 나누는 문제다.

그러기 위해서 우선 fName이라는 포인터에 메모리를 할당하였고, 원본 파일명과 나뉘어진 파일명을 fName에 저장했다.

원본 파일은 x로, 나뉘어진 파일들은 x1/x2/x3로 열었다.

Fstat 함수를 사용하여 원본 파일의 길이를 확인한 후 3등분이 되도록하는 글자 위치를 a, b, c에 저장해줬다. 만약 원본 파일 x의 길이가 129라면 a는 43, b는 86, c는 129라는 값이 저장되고, 그 값으로 x1/x2/x3에 3등분된 크기만큼 쓰도록 만들었다.

```

kyumin@DESKTOP-NUDFAPK ~
$ g++ -o ex2 ex2.c

kyumin@DESKTOP-NUDFAPK ~
$ ./ex2
f1
f1 is split into f11, f12, and f13.
kyumin@DESKTOP-NUDFAPK ~
$ cat f11
I have a dream
that one day this nation
wil
kyumin@DESKTOP-NUDFAPK ~
$ cat f12
l rise up and
live out the true
meaning of
kyumin@DESKTOP-NUDFAPK ~
$ cat f13
its creed
that all men are created equal.

```

```

kyumin@DESKTOP-NUDFAPK ~
$ ls -l
total 641
drwx-----+ 1 kyumin ㄹ ㄹ 0 Mar 15 2021 cdssetup
drwxr-xr-x+ 1 kyumin ㄹ ㄹ 0 Mar 30 01:50 d1
drwxr-xr-x+ 1 kyumin ㄹ ㄹ 0 Mar 6 12:47 d2
-rw-r--r--+ 1 kyumin ㄹ ㄹ 1281 Mar 30 19:10 ex1.c
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ 71584 Mar 30 19:10 ex1.exe
-rw-r--r--+ 1 kyumin ㄹ ㄹ 1002 Mar 30 19:55 ex2.c
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ 69006 Mar 30 19:57 ex2.exe
drwxr-xr-x+ 1 kyumin ㄹ ㄹ 0 Mar 14 13:02 exdir
-rw-r--r--+ 1 kyumin ㄹ ㄹ 129 Mar 30 19:16 f1
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ 43 Mar 30 19:57 f11
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ 43 Mar 30 19:57 f12
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ 43 Mar 30 19:57 f13
-rwxr-xr-x+ 1 kyumin ㄹ ㄹ 129 Mar 25 09:27 f2

```

프로그램을 실행한 후 f1을 입력하였다. 그 파일은 위 사진처럼 3등분되었고, ls -l을 통해 f1파일의 1/3 크기인 43 크기인 것을 확인할 수 있다.

13-1) Modify your code in Problem 13 so that the user can specify the number of small files.

Enter file name to split

f9

How many small files you want to split it into?

5

f9 is split into f91, f92, f93, f94, f95

```
int main(){
    int N,x,y,F[20],S[20]={0};
    char buf[20];
    char *fName[20];

    printf("Enter file name to split\n");
    fName[0]=new char[20];
    scanf("%s",fName[0]);
    x=open(fName[0],O_RDONLY | O_CREAT, 00777);

    printf("How many small files you want to split it into?\n");
    scanf("%d",&N);

    struct stat tmp_stat;
    fstat(x, &tmp_stat);

    int i=1;
    for(i=1;i<=N;i++){
        fName[i]=new char[20];
        sprintf(fName[i],"%s%d",fName[0],i);
        F[i]=open(fName[i],O_RDWR | O_CREAT | O_TRUNC, 00777);
        if(i!=N) S[i]=(int)tmp_stat.st_size/N+S[i-1];
        else S[i]=(int)tmp_stat.st_size;
    }

    int j=0;
    for(i=0;i<N;i++){
        for(;;j++){
            if(S[i]<=j && j<S[i+1]){
                y=read(x, buf, 1);
                if(y==0) break;
                write(F[i+1],buf,y);
            }
            else break;
        }
    }

    printf("%s is split into %s",fName[0],fName[1]);
    for(i=2;i<=N;i++) printf(", %s",fName[i]);

    return 0;
}
```

몇 등분을 할지 사용자의 입력을 받아야하니 N이라는 변수를 만들어 입력을 받았다.

분할된 파일 이름은 fName배열에 저장하기 위해서 sprintf를 사용했다. 여기서 `sprintf(fName[i],"%s%d",fName[0],i);`는 기존 파일 이름에 숫자를 더하고, fName[i]에 저장하라는 의미다.

그리고 분할될 위치를 S변수에 저장해줬다. 원본 파일의 길이를 N으로 나누어 이전 S를 더해 저장해주고, 마지막인 S[N]은 원본의 길이를 저장한다. 만약 131글자의 파일을 3으로 나누어준다면

S[1]=43 S[2]=86 S[3]=131다.

변수가 j인 for문을 만들어서 분할 위치를 저장했던 S를 이용하여 분할된 파일에 한 글자씩 저장해준다.

```
kyumin@DESKTOP-NUDFAPK ~  
$ g++ -g -o ex2 ex2.c  
  
kyumin@DESKTOP-NUDFAPK ~  
$ ./ex2  
Enter file name to split  
f1  
How many small files you want to split it into?  
5  
f1 is split into f11, f12, f13, f14, f15  
kyumin@DESKTOP-NUDFAPK ~  
$ cat f11  
I have a dream  
that one da  
kyumin@DESKTOP-NUDFAPK ~  
$ cat f12  
y this nation  
will rise up  
kyumin@DESKTOP-NUDFAPK ~  
$ cat f13  
and  
live out the true  
me  
kyumin@DESKTOP-NUDFAPK ~  
$ cat f14  
aning of its creed  
that al  
kyumin@DESKTOP-NUDFAPK ~  
$ cat f15  
l men are created equal.
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ ls -l  
total 647  
drwx-----+ 1 kyumin 없음 0 Mar 15 2021 cdssetup  
drwxr-xr-x+ 1 kyumin 없음 0 Mar 30 01:50 d1  
drwxr-xr-x+ 1 kyumin 없음 0 Mar 6 12:47 d2  
-rw-r--r--+ 1 kyumin 없음 1281 Mar 30 19:10 ex1.c  
-rwxr-xr-x+ 1 kyumin 없음 71584 Mar 30 19:10 ex1.exe  
-rw-r--r--+ 1 kyumin 없음 976 Mar 31 00:42 ex2.c  
-rwxr-xr-x+ 1 kyumin 없음 71461 Mar 31 00:42 ex2.exe  
drwxr-xr-x+ 1 kyumin 없음 0 Mar 14 13:02 exdir  
-rw-r--r--+ 1 kyumin 없음 131 Mar 31 00:33 f1  
-rwxr-xr-x+ 1 kyumin 없음 26 Mar 31 00:42 f11  
-rwxr-xr-x+ 1 kyumin 없음 26 Mar 31 00:42 f12  
-rwxr-xr-x+ 1 kyumin 없음 26 Mar 31 00:42 f13  
-rwxr-xr-x+ 1 kyumin 없음 26 Mar 31 00:42 f14  
-rwxr-xr-x+ 1 kyumin 없음 27 Mar 31 00:42 f15  
-rwxr-xr-x+ 1 kyumin 없음 129 Mar 25 09:27 f2  
-rwxr-xr-x+ 1 kyumin 없음 18 Mar 27 00:16 f3
```

프로그램을 실행해서 f1을 5등분했다. Cat을 통해 분할된 파일의 내용을 확인해보면 정상적으로 분할된 것을 볼 수 있고, ls -l을 이용하여 크기를 보면 5등분이 된 것을 볼 수 있다.

14) What is wrong with following program?

```
char temp0[20], *temp1[10], *temp2[10];  
printf( "enter src file name\n" );  
gets(temp0);  
temp1[0]=temp0;  
printf( "enter dest file name\n" );  
gets(temp0);  
temp2[0]=temp0;  
printf( "src file:%s dest file:%s\n" , temp1[0], temp2[0]);
```

temp1과 temp2는 포인터고, 동적 메모리 할당이 되지 않았다. 그래서 temp1[0]=temp0;는 단순히 temp0의 스트링의 주소를 가르킬 뿐이다. Temp1과 temp2가 가리키는 주소는 temp0으로 동일하다. 그래서 temp1과 temp2를 출력해보면 동일한 결과가 나올 뿐이다. 이 문제를 해결하기 위해서는 포인터에 메모리 할당을 한 후 스트링을 저장하거나 temp0이 아닌 다른 캐릭터 배열을 만들어 입력을 따로 받으면 문제를 해결할 수 있다.

```
int x, x1, y;
x=open( "f1" , O_RDONLY, 00777);
x1=open( "f2" , O_WRONLY|O_CREAT|O_TRUNC,00777);
char buf[512];
int cnt=0;
for(;;){
    y=read(x,buf,1);
    if (y==0) break;
    cnt++;
}
write(x1, buf, cnt);
```

F1의 길이인 1310 cnt에 저장된 것을 볼 수 있다.