

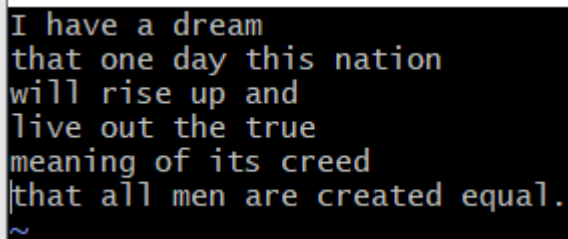
Week4 HW

12201922 이규민

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

```
$vi f1
```

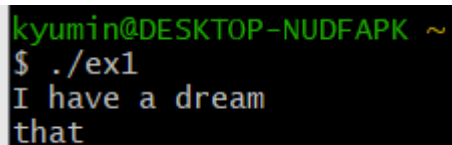
```
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.
```



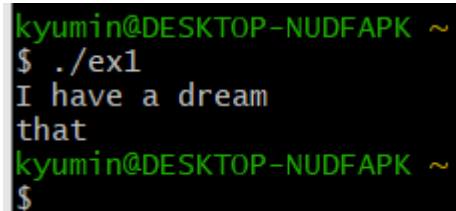
```
I have a dream
that one day this nation
will rise up and
live out the true
meaning of its creed
that all men are created equal.
~
```

Vi f1으로 코맨드 모드에 진입하여 내용을 입력하였다.

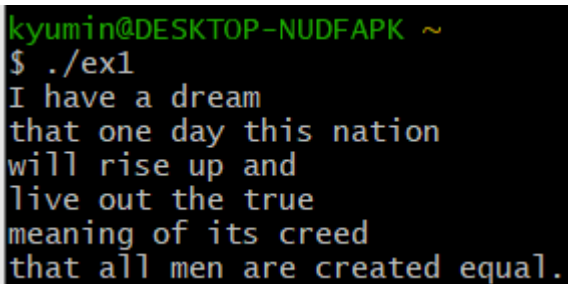
2) Try the code in 6-0), 6-1), 6-2), 6-3), 6-4), 6-5). For 6-3) explain the strange output.



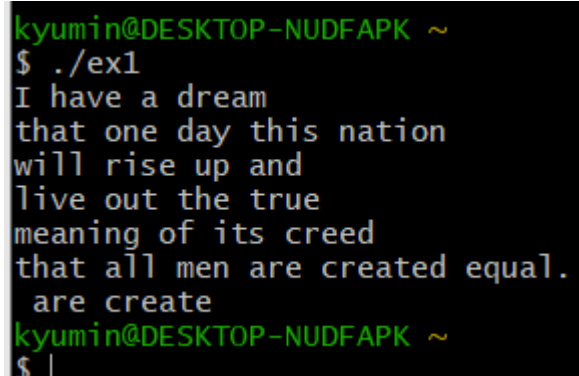
```
kyumin@DESKTOP-NUDFAPK ~
$ ./ex1
I have a dream
that
```



```
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 y가 아닌 20이라고 작성했기 때문이다.

Buf 배열은 저장되어있는 스트링보다 저장할 스트링의 길이가 짧다면 이전 문자열이 기록에 남는다. 즉 기존에 “t 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

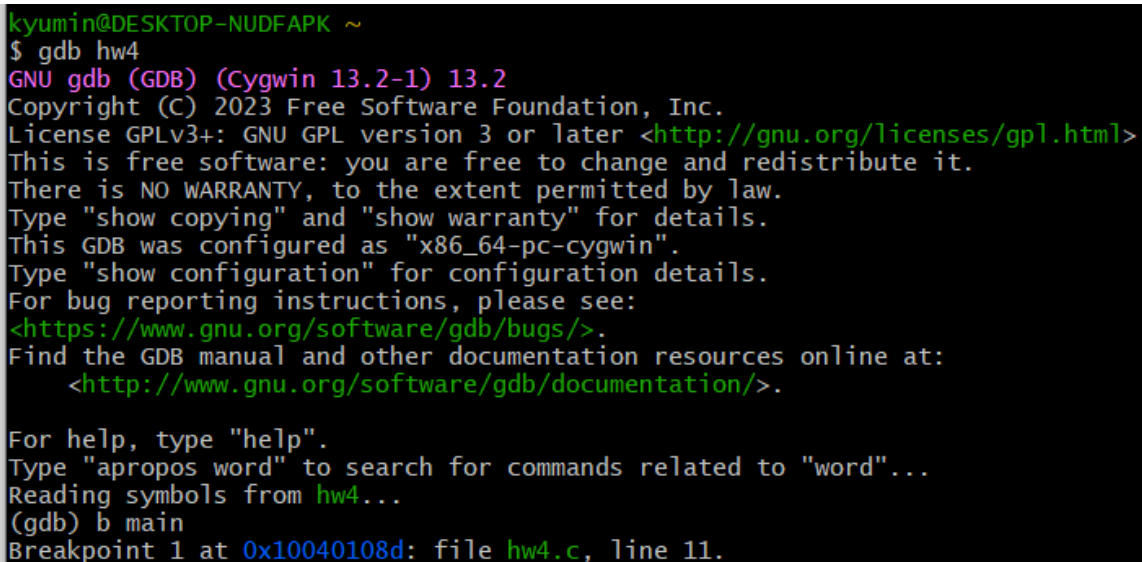
gdb) list                              -- show rest of code

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 main이라고 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 2월 10 0 Mar 15 2021 cdssetup  
-rw-r--r--+ 1 kyumin 2월 10 1945 Mar 24 00:15 char1.c  
-rwxr-xr-x+ 1 kyumin 2월 10 68715 Mar 23 23:55 char1.exe  
-rwxr-xr-x+ 1 kyumin 2월 10 430 Mar 27 10:02 cphw4.c  
drwxr-xr-x+ 1 kyumin 2월 10 0 Mar 9 22:13 d1  
drwxr-xr-x+ 1 kyumin 2월 10 0 Mar 6 12:47 d2  
-rw-r--r--+ 1 kyumin 2월 10 376 Mar 27 10:01 ex1.c  
-rwxr-xr-x+ 1 kyumin 2월 10 66197 Mar 27 10:01 ex1.exe  
-rwxr-xr-x+ 1 kyumin 2월 10 65816 Mar 14 23:01 ex2.exe  
-rw-r--r--+ 1 kyumin 2월 10 221 Mar 16 16:40 ex3.c  
-rwxr-xr-x+ 1 kyumin 2월 10 66407 Mar 16 16:41 ex3.exe  
-rw-r--r--+ 1 kyumin 2월 10 54 Mar 14 12:57 ex4.c  
drwxr-xr-x+ 1 kyumin 2월 10 0 Mar 14 13:02 exdir  
-rw-r--r--+ 1 kyumin 2월 10 129 Mar 25 09:18 f1  
-rwxr-xr-x+ 1 kyumin 2월 10 129 Mar 25 09:27 f2  
-rwxr-xr-x+ 1 kyumin 2월 10 18 Mar 27 09:46 f3  
-rw-r--r--+ 1 kyumin 2월 10 430 Mar 25 10:45 hw4.c  
-rwxr-xr-x+ 1 kyumin 2월 10 66926 Mar 26 14:55 hw4.exe  
-rw-r--r--+ 1 kyumin 2월 10 279742 Mar 14 12:51 x  
-rw-r--r--+ 1 kyumin 2월 10 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.

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.

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
```

```
.....
```

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

.....

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);
```

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);
```

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.

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

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]);
```

15) What is wrong with following program. Find the problem with GDB and fix it.

```
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);
```

