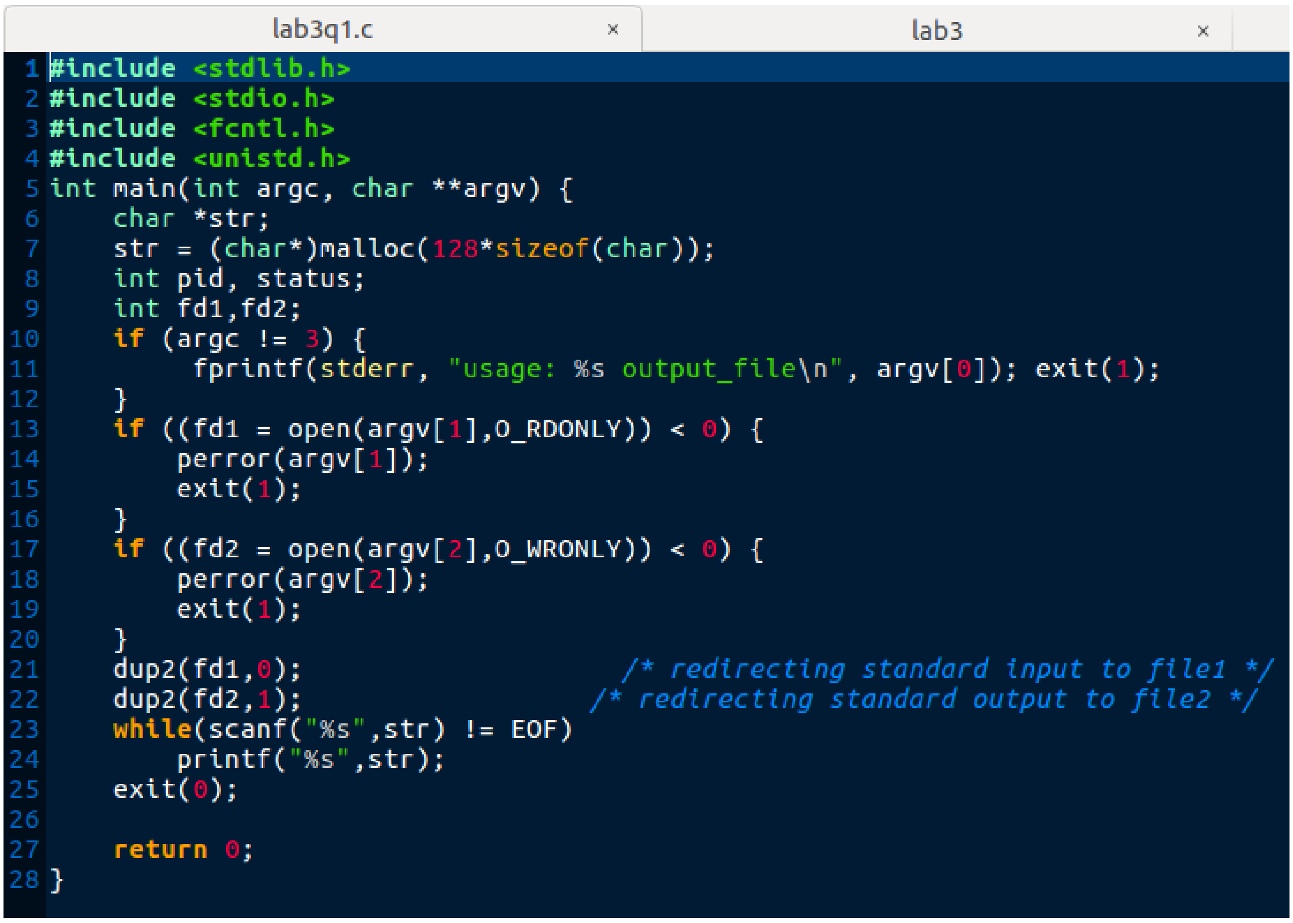
# Advanced Unix Programming
# Assignment-3

**Group Members:**
**Vijesh Ghandare 111403013**
**Nikhil Gawande 111408013**
**Anupam Godse 111408016**

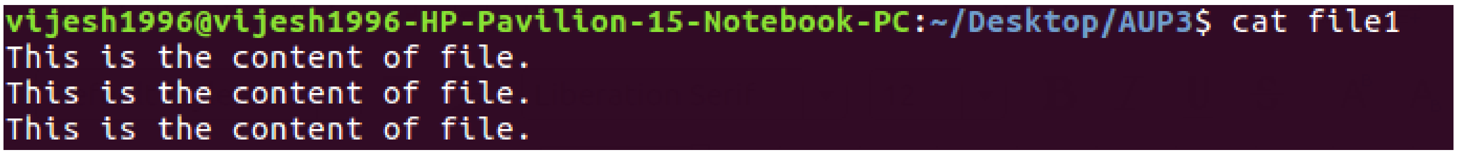
**Q1.Using dup function redirect stdin to file1 and stdout to file2. Read a line using scanf and write the same using printf. Verify the contents of both files.**

**CODE:**



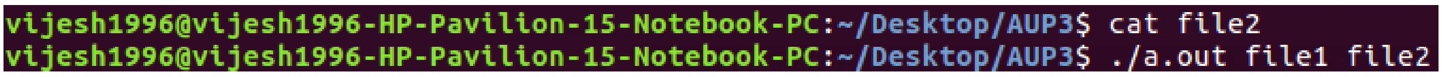
```c
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, char **argv) {
    char *str;
    str = (char*)malloc(128*sizeof(char));
    int pid, status;
    int fd1,fd2;
    if (argc != 3) {
         fprintf(stderr, "usage: %s output_file\n", argv[0]); exit(1);
    }
    if ((fd1 = open(argv[1],O_RDONLY)) < 0) {
        perror(argv[1]);
        exit(1);
    }
    if ((fd2 = open(argv[2],O_WRONLY)) < 0) {
        perror(argv[2]);
        exit(1);
    }
    dup2(fd1,0);                    /* redirecting standard input to file1 */
    dup2(fd2,1);                  /* redirecting standard output to file2 */
    while(scanf("%s",str) != EOF)
        printf("%s",str);
    exit(0);

    return 0;
}
```

**Outputs:**
**Initial content of file1:**



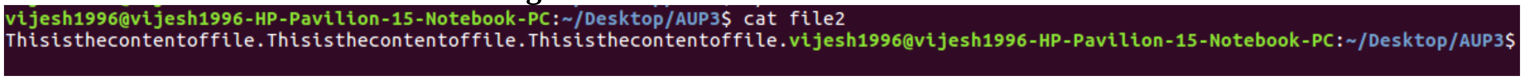
```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cat file1
This is the content of file.
This is the content of file.
This is the content of file.
```

**Initially file2 is empty:**



```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cat file2
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out file1 file2
```

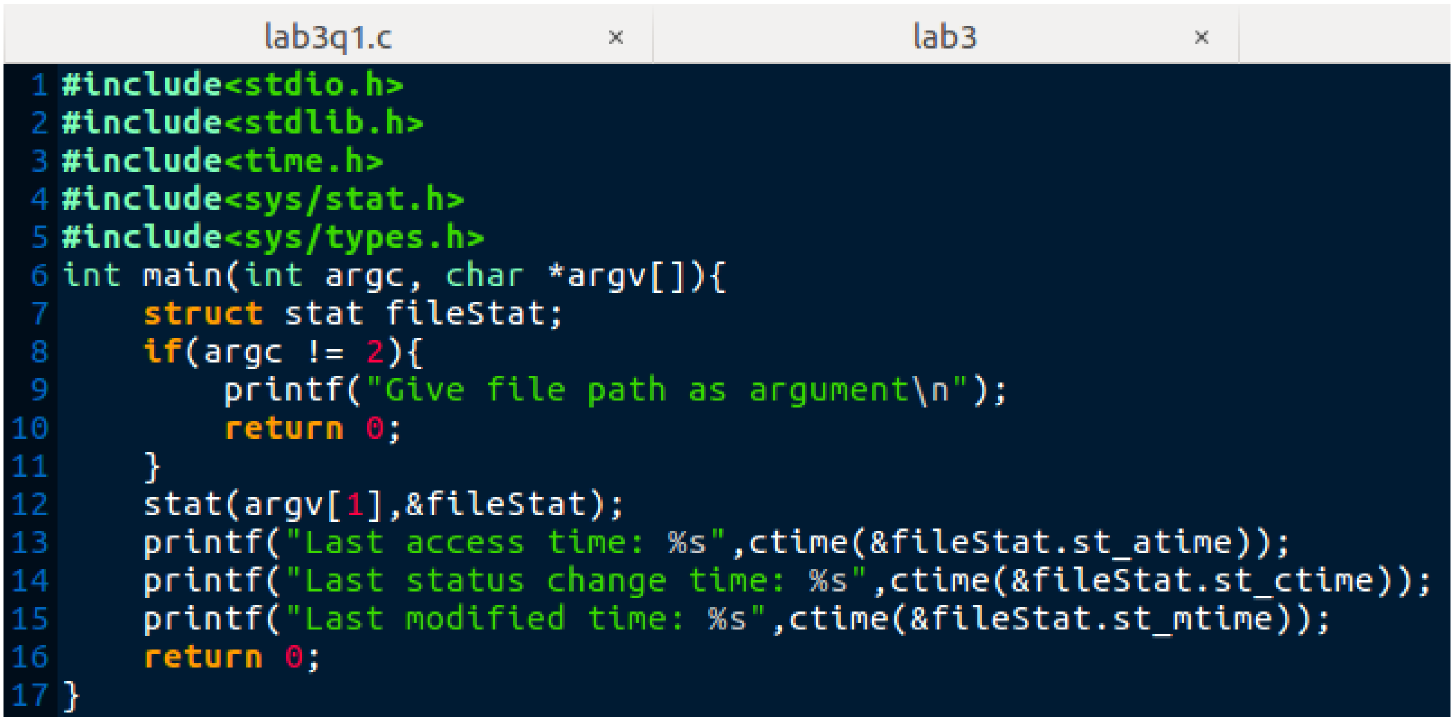
**Content of file2 after executing code:**



```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cat file2
Thisisthecontentoffile.Thisisthecontentoffile.Thisisthecontentoffile.vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$
```

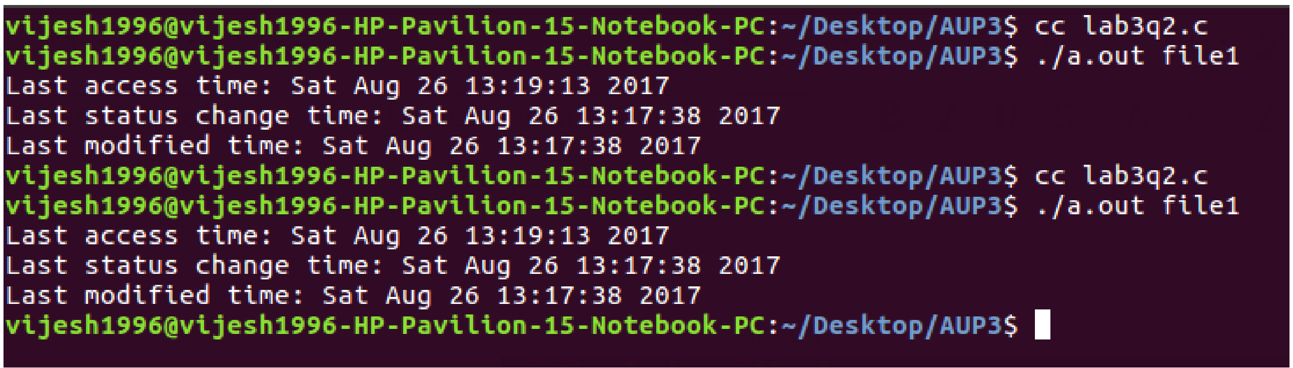
**Explaination:**
So, The contents of both files are same but the only difference is the content of file2 is not newline separated as scanf() function doesn't read newline character so that will not be written in the file2.

**Q2: Does calling stat function change any of the time values? Verify with a program.**
**Code:**



```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/stat.h>
#include<sys/types.h>
int main(int argc, char *argv[]){
    struct stat fileStat;
    if(argc != 2){
        printf("Give file path as argument\n");
        return 0;
    }
    stat(argv[1],&fileStat);
    printf("Last access time: %s",ctime(&fileStat.st_atime));
    printf("Last status change time: %s",ctime(&fileStat.st_ctime));
    printf("Last modified time: %s",ctime(&fileStat.st_mtime));
    return 0;
}
```

**Output:**

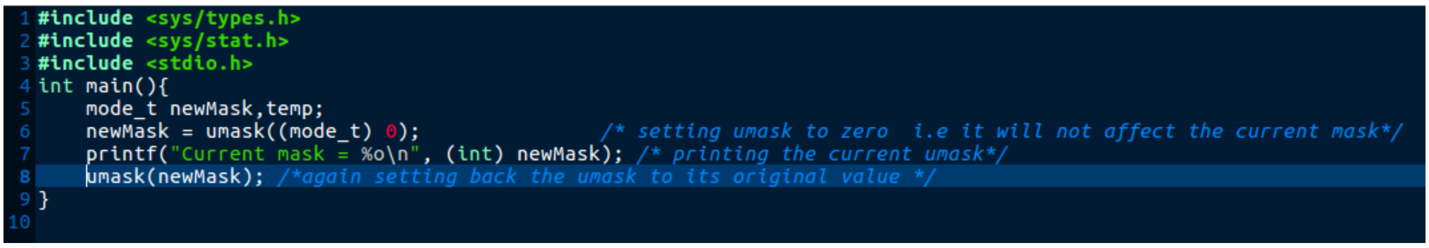


```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cc lab3q2.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out file1
Last access time: Sat Aug 26 13:19:13 2017
Last status change time: Sat Aug 26 13:17:38 2017
Last modified time: Sat Aug 26 13:17:38 2017
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cc lab3q2.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out file1
Last access time: Sat Aug 26 13:19:13 2017
Last status change time: Sat Aug 26 13:17:38 2017
Last modified time: Sat Aug 26 13:17:38 2017
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$
```

**Explaination:**
A file has three timestamps:
1)st_atime: Time of last file access
2)st_mtime: Time of last file modification
3)st_ctime: Time of last status change
Accessing a file's metadata via stat() doesn’t change any of these entries. stat() just access information in the inode, not in the file itself.
This is confirmed by the results of this code by running it twice on the same file.

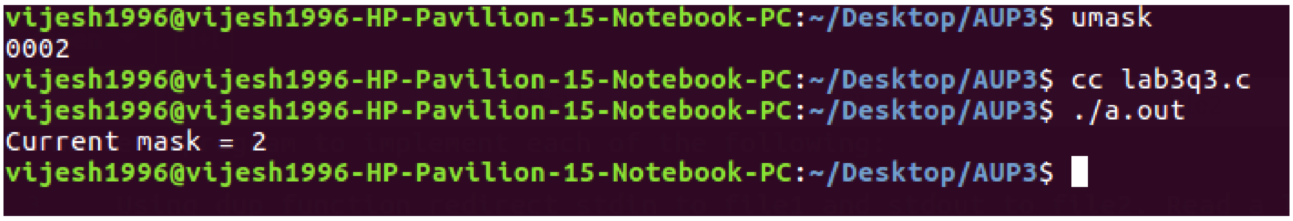
---

**Q3. umask() always sets the process umask and, at the same time, returns a copy of the old umask. How can we obtain a copy of the current process umask while leaving it unchanged? Write a program to demonstrate.**
**Code:**



```c
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
int main(){
    mode_t newMask,temp;
    newMask = umask((mode_t) 0);               /* setting umask to zero  i.e it will not affect the current mask*/
    printf("Current mask = %o\n", (int) newMask); /* printing the current umask*/
    umask(newMask); /*again setting back the umask to its original value */
}
```

**Output:**



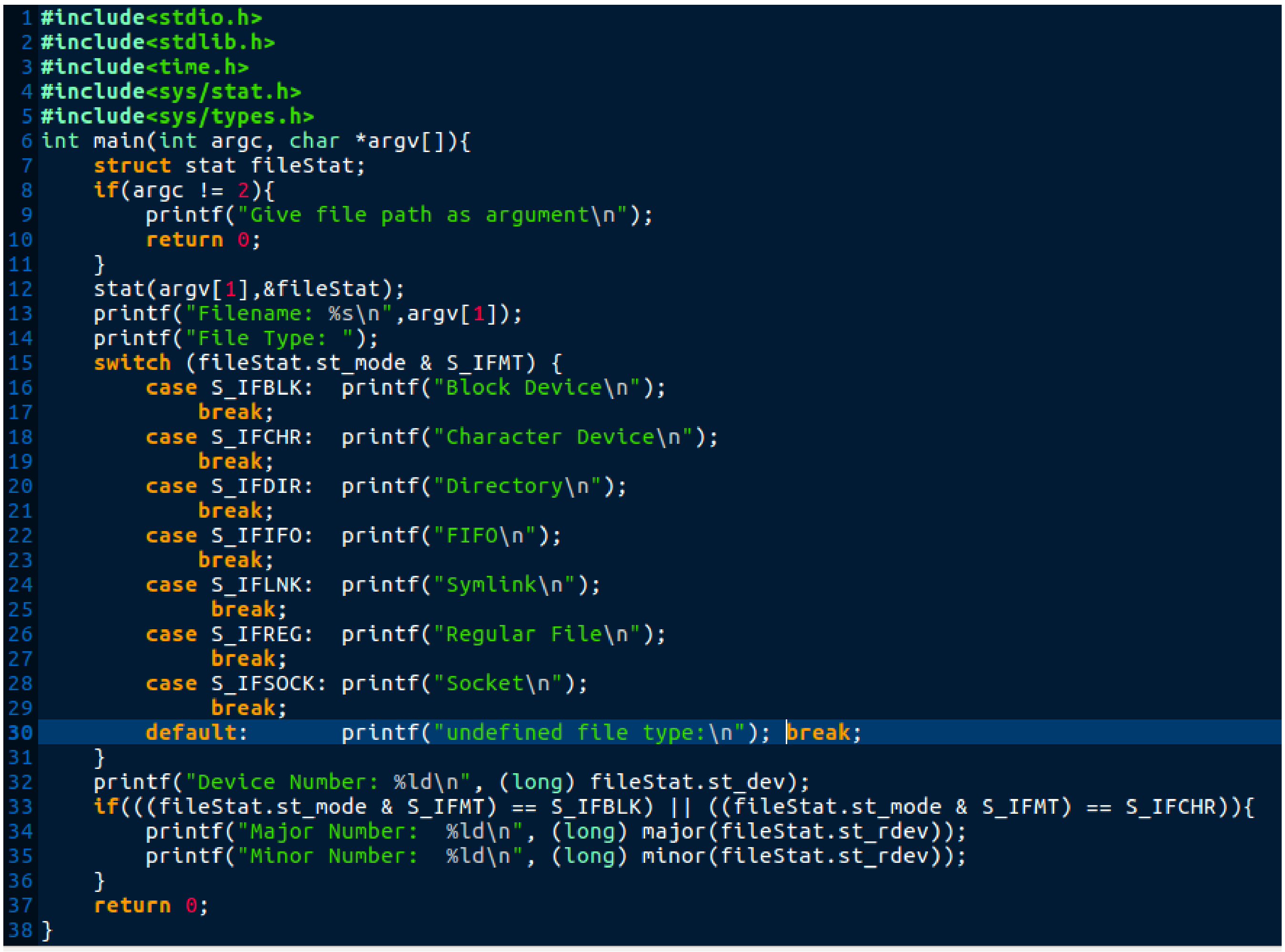
```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ umask
0002
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cc lab3q3.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out
Current mask = 2
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ 
```

**Explaination:**
As per the code setting the unmask to zero which does not affect the current file permissions and storing current umask in the variable.
The value in the variable is our required umask of current process.
Print the current value and set the umask to its original value.

---

**Q4: Display the device number for the filename input as command line argument. If it is a character or block special file, then display its major and minor numbers.**

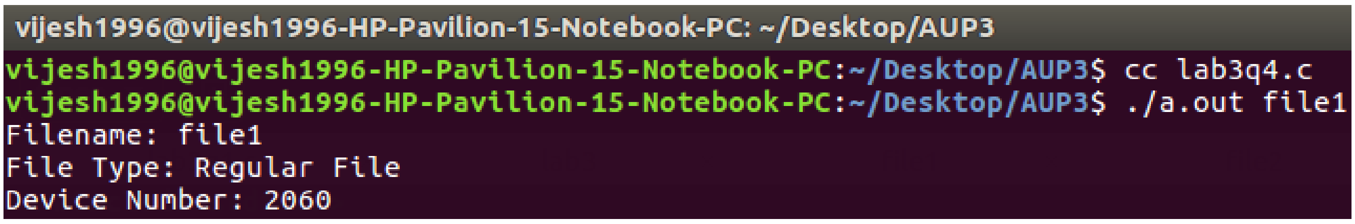
**Code:**



```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/stat.h>
#include<sys/types.h>
int main(int argc, char *argv[]){
    struct stat fileStat;
    if(argc != 2){
        printf("Give file path as argument\n");
        return 0;
    }
    stat(argv[1],&fileStat);
    printf("Filename: %s\n",argv[1]);
    printf("File Type: ");
    switch (fileStat.st_mode & S_IFMT) {
        case S_IFBLK:  printf("Block Device\n");
            break;
        case S_IFCHR:  printf("Character Device\n");
            break;
        case S_IFDIR:  printf("Directory\n");
            break;
        case S_IFIFO:  printf("FIFO\n");
            break;
        case S_IFLNK:  printf("Symlink\n");
             break;
        case S_IFREG:  printf("Regular File\n");
             break;
        case S_IFSOCK: printf("Socket\n");
             break;
        default:       printf("undefined file type:\n"); break;
    }
    printf("Device Number: %ld\n", (long) fileStat.st_dev);
    if(((fileStat.st_mode & S_IFMT) == S_IFBLK) || ((fileStat.st_mode & S_IFMT) == S_IFCHR)){
        printf("Major Number:  %ld\n", (long) major(fileStat.st_rdev));
        printf("Minor Number:  %ld\n", (long) minor(fileStat.st_rdev));
    }
    return 0;
}
```

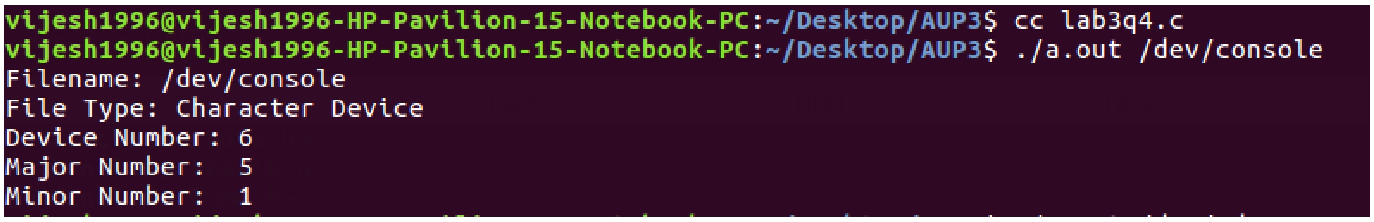
**Outputs:**

**Case I): For files other than Block special files and Character special files:**



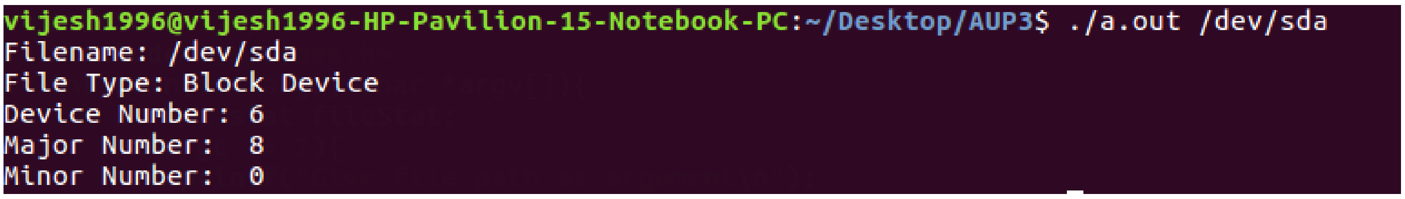
```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC: ~/Desktop/AUP3
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cc lab3q4.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out file1
Filename: file1
File Type: Regular File
Device Number: 2060
```

**Case II: For Character Device files:**



```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ cc lab3q4.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out /dev/console
Filename: /dev/console
File Type: Character Device
Device Number: 6
Major Number:  5
Minor Number:  1
```

**Case III: For Block Device Files:**



```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/AUP3$ ./a.out /dev/sda
Filename: /dev/sda
File Type: Block Device
Device Number: 6
Major Number:  8
Minor Number:  0
```