

Q1)“The child “exec” call inherits the file descriptors of parent if Close\_on\_exec is not set“. Demonstrate with an example

Code for the parent pocess :

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

int main(){
    int fd, pid;
    fd = open("input.txt", O_RDONLY);
    int n = fcntl(fd,F_SETFD,FD_CLOEXEC);
    if((pid = fork()) < 0) {
        perror("fork");
        exit(1);
    }
    char buf[10];
    sprintf(buf,"%d",fd);
    if(pid == 0)
        execl("child","child",buf, NULL);
    return 0;
}
```

Code for execed process :

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<fcntl.h>
#include<errno.h>
#include<string.h>

extern int errno;

int main(int argc, char *argv[]){
    int fd, pid;
    char buf[6];
    fd = atoi(argv[1]);    /*File descriptor opened in parent*/
    int n = read(fd, buf, 5);
    buf[5] = '\0';
    if(n == -1){
        fprintf(stderr, "Error opening file : %s\n", strerror(errno));
    }
    else
        printf("%s\n",buf);
    return 0;
}
```

Execution :

```
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7
File Edit View Search Terminal Help
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ cc q1.c -o q1
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ cc q1_child.c -o child
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ cat q1_input.txt
1. Threads require less program and system overhead to run than processes do. The operating
system performs less work on behalf of a multithreaded program than it does for a multiprocess
program. This translates into a performance gain for the multithreaded program.
2. You create a new process by using the UNIX fork system call, you create a new thread by calling
the pthread_create Pthreads function
3. Threads don't have a parent/child relationship as processes do. So, any thread can cancel any
other thread, as long as the canceling thread has the thread handle of its victim. Because you
want your application to be solidly structured, you'll cancel threads only from the thread that
initially created them. The only thread that has slightly different properties than any other is the
first thread in the process, which is known as the main thread.
4. Wait = pthread_join
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ ./q1 child
Error opening file : Bad file descriptor
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $
```

When the file is opened using close on exec flag set, the child process which does an exec cannot play around with this file descriptor. These file open descriptors are not accessible in the execed process. The execed process in this case is named as “child”.

q1.c is the main process where the file has been opened with FD\_CLOEXEC flag set using fcntl.

Q2) Write a program that takes a file name as an argument, opens the file, reads it and closes the file. The file should contain a string with the name of another application (e.g., 'ls' or 'ps' or any of your own applications) and the program forks a new process that executes the application named in the file.

Code:

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

int main(int argc, char *argv[]) {
    int fd, i, ret, rete;
    char prog[256];
    char *arg[2];
    pid_t child;
    if(argc != 2) {
        printf("Usage:\nnash <filename>\n");
        return 1;
    }

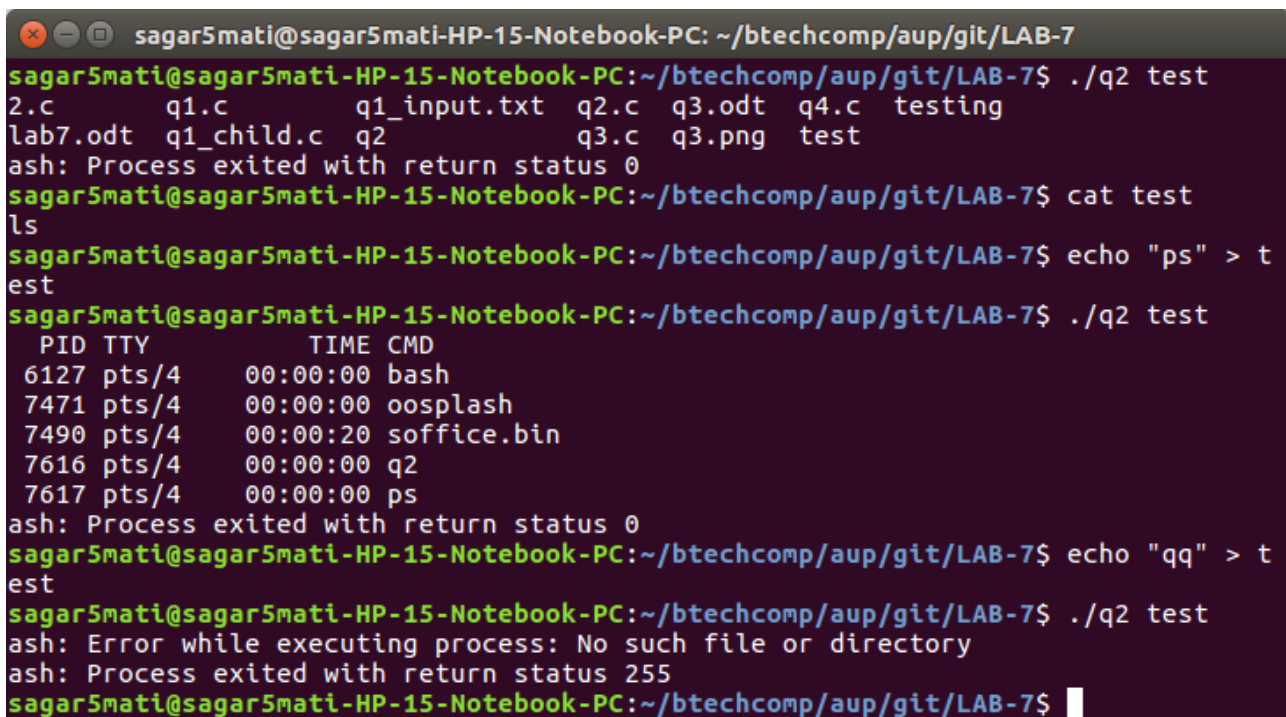
    if((fd = open(argv[1], O_RDONLY)) == -1) {
        perror("ash: Unable to open file");
        return -1;
    }
}
```

```

i = 0;
while(read(fd, &prog[i++], sizeof(char)));
prog[i - 2] = '\0';
arg[0] = prog;
arg[1] = NULL;
if((child = vfork()) < 0) {
    perror("ash: Unable to fork new process");
    return -1;
}
if(child == 0) {
    if((rete = execvp(prog, arg)) == -1) {
        perror("ash: Error while executing process");
        _exit(-1);
    }
}
else {
    wait(&ret);
    if(WIFEXITED(ret)) {
        printf("ash: Process exited with return status %d\n", WEXITSTATUS(ret));
    }
}
return 0;
}

```

Execution:



```

sagar5mati@sagar5mati-HP-15-Notebook-PC: ~/btechcomp/aup/git/LAB-7
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$ ./q2 test
2.c      q1.c      q1_input.txt  q2.c  q3.odt  q4.c  testing
lab7.odt q1_child.c q2      q3.c  q3.png  test
ash: Process exited with return status 0
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$ cat test
ls
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$ echo "ps" > test
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$ ./q2 test
  PID TTY          TIME CMD
 6127 pts/4    00:00:00 bash
 7471 pts/4    00:00:00 oosplash
 7490 pts/4    00:00:20 soffice.bin
 7616 pts/4    00:00:00 q2
 7617 pts/4    00:00:00 ps
ash: Process exited with return status 0
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$ echo "qq" > test
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$ ./q2 test
ash: Error while executing process: No such file or directory
ash: Process exited with return status 255
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-7$

```

Q3) Implement cat < hw.txt > hw-copy.txt

Code :

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

int main(int argc, char *argv[]) {
    int fd1, fd2, num;
    fd1 = open(argv[1], O_RDONLY);
    fd2 = open(argv[2], O_WRONLY|O_CREAT|O_APPEND|O_TRUNC, 0644);

    // hw.txt becomes the standard input
    if(dup2(fd1, 0) == -1) {
        printf("Error\n");
        return -1;
    }
    // integer is taken from hw.txt instead of standard input
    scanf("%d", &num);

    // hw-copy.txt becomes the standard output
    if(dup2(fd2, 1) == -1) {
        printf("Error\n");
        return -1;
    }

    printf("This will go to hw-copy.txt and not standard output\n");
    printf("This is this integer taken as input from hw.txt %d\n", num);
    return 0;
}
```

Execution :

```
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7
File Edit View Search Terminal Help
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ cc q3.c -o q3
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ ./q3 hw.txt hw-copy.txt
111403076
hello world
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ cat q3.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/wait.h>

int main(int argc, char *argv[]) {
    int fd1, fd2, num;
    fd1 = open(argv[1], O_RDONLY);
    fd2 = open(argv[2], O_WRONLY|O_CREAT|O_APPEND|O_TRUNC, 0644);

    // hw.txt becomes the standard input
    if(dup2(fd1, 0) == -1) {
        printf("Error\n");
        return -1;
    }

    // integer is taken from hw.txt instead of standard input
    scanf("%d", &num);

    // hw-copy.txt becomes the standard output
    if(dup2(fd2, 1) == -1) {
        printf("Error\n");
        return -1;
    }
    printf("This will go to hw-copy.txt and not standard output\n");
    printf("This is this integer taken as input from hw.txt %d\n", num);
    return 0;
}
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $ cat hw-copy.txt
This will go to hw-copy.txt and not standard output
This is this integer taken as input from hw.txt 111403076
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-7 $
```

Here fd1 is the file descriptor for file hw.txt. fd2 is the file descriptor for file hw-copy.txt

dup2 function is used here to duplicate file descriptors.

hw.txt acts as standard input (STDIN – 0). Therefore the integer taken as input is taken from hw.txt rather than STDIN.

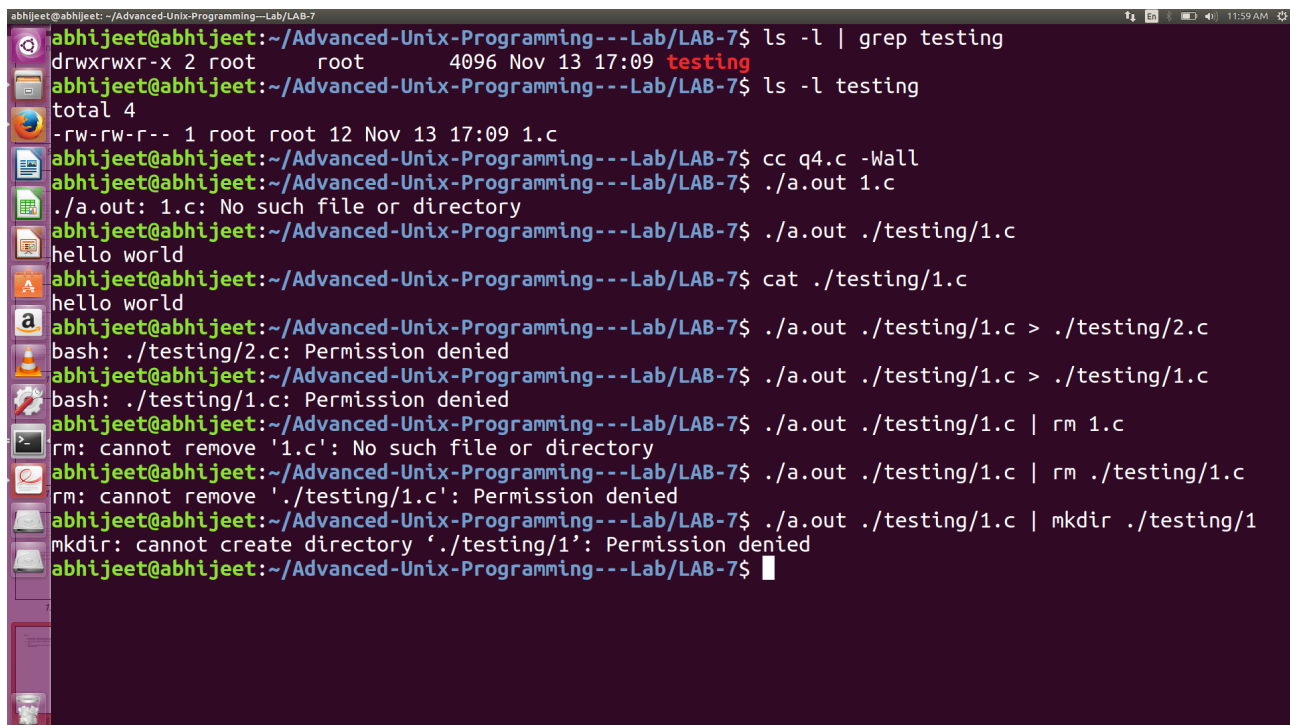
Hw-copy.txt acts as standard output (STDOUT – 1). Therefore whatever we print from this file we write into hw-copy.txt instead of STDOUT.

Q4) Bob works for an auditing agency needs to be able to read all the files in the system. The system admin has to protect the integrity of the system and should not allow Bob to modify or delete any file. Write a special SETUID program for the admin so that he can gave the executable permission of it to Bob. This program requires Bob to type a file name at the command line and then it will run /bin/cat to display the specified file. Can Bob compromise the integrity (by adding/modifying/deleting files) of this system? How?

Code:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main(int argc, char *argv[]) {
    if(argc < 2) {
        printf("Arguments are insufficient\n");
        return 1;
    }
    setuid(2000);
    execv("/bin/cat", argv);
    return 0;
}
```

Execution:



```
abhiyeet@abhiyeet: ~/Advanced-Unix-Programming---Lab/LAB-7
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ls -l | grep testing
drwxrwxr-x 2 root root 4096 Nov 13 17:09 testing
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ls -l testing
total 4
-rw-rw-r-- 1 root root 12 Nov 13 17:09 1.c
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ cc q4.c -Wall
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out 1.c
./a.out: 1.c: No such file or directory
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out ./testing/1.c
hello world
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ cat ./testing/1.c
hello world
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out ./testing/1.c > ./testing/2.c
bash: ./testing/2.c: Permission denied
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out ./testing/1.c > ./testing/1.c
bash: ./testing/1.c: Permission denied
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out ./testing/1.c | rm 1.c
rm: cannot remove '1.c': No such file or directory
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out ./testing/1.c | rm ./testing/1.c
rm: cannot remove './testing/1.c': Permission denied
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$ ./a.out ./testing/1.c | mkdir ./testing/1
mkdir: cannot create directory './testing/1': Permission denied
abhiyeet@abhiyeet:~/Advanced-Unix-Programming---Lab/LAB-7$
```

In the above screenshot we can see that there exists a directory namely testing. The owner and the group owner of this directory is root. There exists a file 1.c inside testing with the root as owner and group owner. When we compile and run our program to read this 1.c, it executes and reads the data "hello world" from 1.c. As we can see above no other operation can be performed such that a file can be added /modified or deleted from this directory. This happens because a call to setuid function has been made with uid = 2000. The current owner of the directory and the file is root and the real uid of the process is 1000 (myuid). So with this program the system admin can ensure integrity of the system.