---

1. **"The child "exec" call inherits the file descriptors of parent if Close_on_exec is not set".**
Demonstrate with an example.

Program:-

```
                              l7q1.c                                    ×
 1 #include <unistd.h>
 2 #include <sys/types.h>
 3 #include <sys/stat.h>
 4 #include <fcntl.h>
 5 #include <stdio.h>
 6 #include <stdlib.h>
 7
 8 int main() {
 9       int fd;
10       pid_t pid;
11       fd = open("testfile", O_RDWR | O_CREAT, 0777);
12       if(fd == -1) {
13             printf("open failed\n");
14       }
15       write(fd, "This is a line added by parent.\n", 32);
16       //we will check default status of close on exec flag
17       int val = fcntl(fd, F_GETFD);
18       if(val) {
19             printf("close on exec is set :- %d\n", val);
20       }
21       else {
22             printf("close on exec is not set\n");
23       }
24       if((pid = fork()) < 0) {
25             printf("fork failed\n");
26       }
27       else if(pid == 0) {
28             char buff[20];
29             snprintf(buff, 20, "%d", fd);
30             printf("%s\n", buff);
31             if(execl("progforexec", "progforexec", buff, (char*)NULL) == -1) {
32                   printf("exec failed\n");
33             }
34       }
35       return 0;
36 }
37
```

Program used for execution in child :-

```
 1 #include <unistd.h>
 2 #include <sys/types.h>
 3 #include <sys/stat.h>
 4 #include <fcntl.h>
 5 #include <stdio.h>
 6 #include <string.h>
 7 #include <stdlib.h>
 8
 9 int main(int argc, char *argv[]) {
10     int fd = atoi(argv[1]);
11     if(write(fd, "This is line written by program executed in child using exec().\n",
12             strlen("This is line written by program executed in child using exec().\n")) < 0) {
13         printf("Demonstration failed\n");
14     }
15     else {
16         printf("The child "exec" call inherits the file descriptors of parent if Close_on_exec is not set : Demonstation successfull\n");
17         exit(0);
18     }
19     return 0;
20 }
```

Directory contents:-

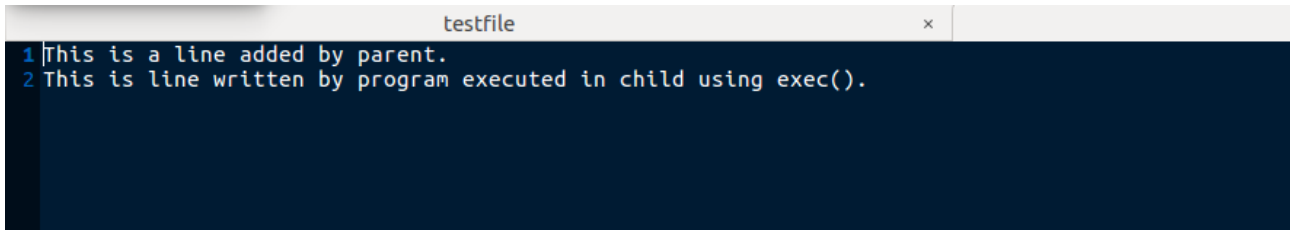a   l7q1.c   progforexec   progforexec.c   testfile
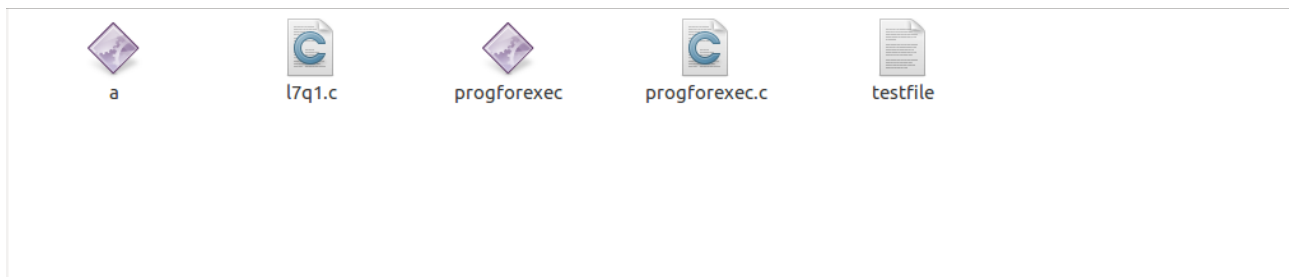
Execution:-

```
anupam@anupam-Inspiron-7548:~/aup$ cc l7q1.c -o a
anupam@anupam-Inspiron-7548:~/aup$ ./a
close on exec is not set
3
The child "exec" call inherits the file descriptors of parent if Close_on_exec is not set : Demonstation successfull
anupam@anupam-Inspiron-7548:~/aup$
```

Before exec call:-

testfile   ×

```
1 This is a line added by parent.
```
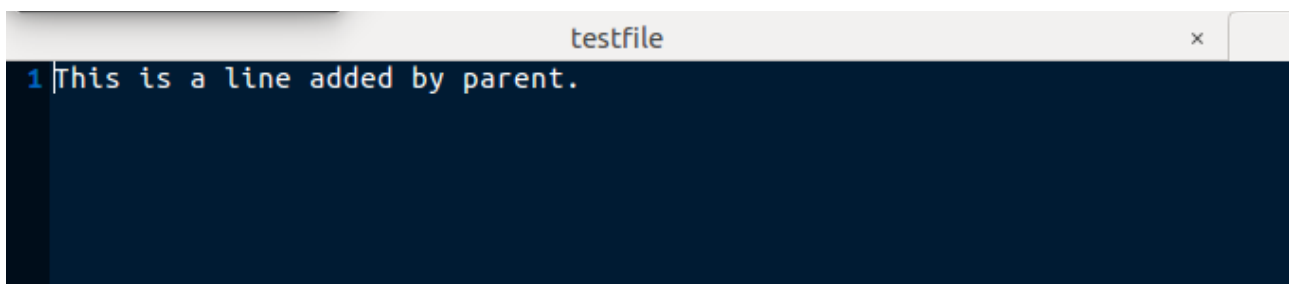
After exec call:-


```
testfile                                          ×
1 This is a line added by parent.
2 This is line written by program executed in child using exec().
```

Explanation:-

As close on exec flag is not set all the file descriptors open in parent are inherited by child exec() call. This can be demonstrated by,
1) Creating a new file in parent and writing some content to the file.
2) Passing the file descriptor as an argument to the program to be executed.
3) Using the same descriptor in the program that is executed using exec to see if it can be used to write/read from the same file.

As demonstrated in above program output, we can see that the file was created by parent program and the file descripted was passed to the program to be executed. The program to be executed used the file descriptor to write some content in file which was successfull. Hence we can conclude that, "The child "exec" call inherits the file descriptors of parent if Close_on_exec is not set".

2.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.

Program:-

```c
1 #include <unistd.h>
2 #include <sys/types.h>
3 #include <sys/stat.h>
4 #include <fcntl.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8 #include <sys/wait.h>
9
10
11 int main(int argc, char *argv[]) {
12     int fd, count = 0;;
13     pid_t pid;
14     char *arg[128];
15     fd = open("testfile", O_RDONLY);
16     if(fd == -1) {
17         printf("Open failed\n");
18     }
19     char buff[128];
20     read(fd, buff, 128);
21     printf("%s\n", buff);
22     char *token = strtok(buff, " \n");
23     while(token) {
24         arg[count++] = token;
25         printf("%s\n", token);
26         token = strtok(NULL, " \n");
27     }
28     arg[count] = NULL;
29     if((pid = fork()) < 0) {
30         printf("fork failed\n");
31     }
32     else if(pid == 0) {
33         execvp(arg[0], arg);
34     }
35     else {
36         wait(NULL);
37     }
38     return 0;
39 }
```

Testfile :-

```
1 ls -l
```

Output :-

```
anupam@anupam-Inspiron-7548:~/aup/lab7/q2$ cc l7q2.c -o a
anupam@anupam-Inspiron-7548:~/aup/lab7/q2$ ./a
total 88
-rwxrwxr-x 1 anupam anupam  8968 Oct  7 18:00 a
-rw-rw-r-- 1 anupam anupam   695 Oct  7 18:00 l7q2.c
-rw-rw-r-- 1 anupam anupam 63928 Oct  7 17:49 l7q2.png
-rw-rw-r-- 1 anupam anupam  1234 Oct  7 17:50 l7q2_testfile.png
-rw-rw-r-- 1 anupam anupam     6 Oct  7 17:40 testfile
anupam@anupam-Inspiron-7548:~/aup/lab7/q2$
```

3.   Implement cat < hw.txt > hw-copy.txt

Program:-

```c
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/wait.h>
5 #include <sys/stat.h>
6 #include <fcntl.h>
7 #include <errno.h>
8
9 int main(int argc, char *argv[]){
10      int fdr = open(argv[1], O_RDONLY);
11      int fdw = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0777);
12      pid_t pid;
13      if((pid = fork()) < 0) {
14              printf("fork failed\n");
15      }
16      else if(pid == 0){
17              dup2(fdr,0);
18              dup2(fdw,1);
19              execl("/bin/sh","sh", "-c", "cat", (char*)0);
20              _exit(127);
21      }
22      if(waitpid(pid, NULL, 0) != pid) {
23              return errno;
24      }
25      return 0;
26 }
```

Execution :-

```
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ ls
copyfrom  copyto  l7q3.c  l7q3.png
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ cat copyfrom
content of copyfrom.
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ cat copyto
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ cc l7q3.c -o a
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ ./a copyfrom copyto
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ cat copyfrom
content of copyfrom.
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ cat copyto
content of copyfrom.
anupam@anupam-Inspiron-7548:~/aup/lab7/q3/new$ █
```

4.   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?