# Assignment 7

**Niramay Vaidya 111605075**
**Srishti Shelke 111603056**

**1.** Create a game program that switches between the effective user ID and real user ID. The game player may write details (like game iteration number) to a file owned by the game player and manipulates a scores file that should be writable only by the game program owner. Both the game program and scores file are owned by the game program owner. Demonstrate that the game player can switch between the files in turns as own file, scores file, own file and scores file.

The number at the beginning is the iteration number followed by the current user accessing the file and its corresponding euid.

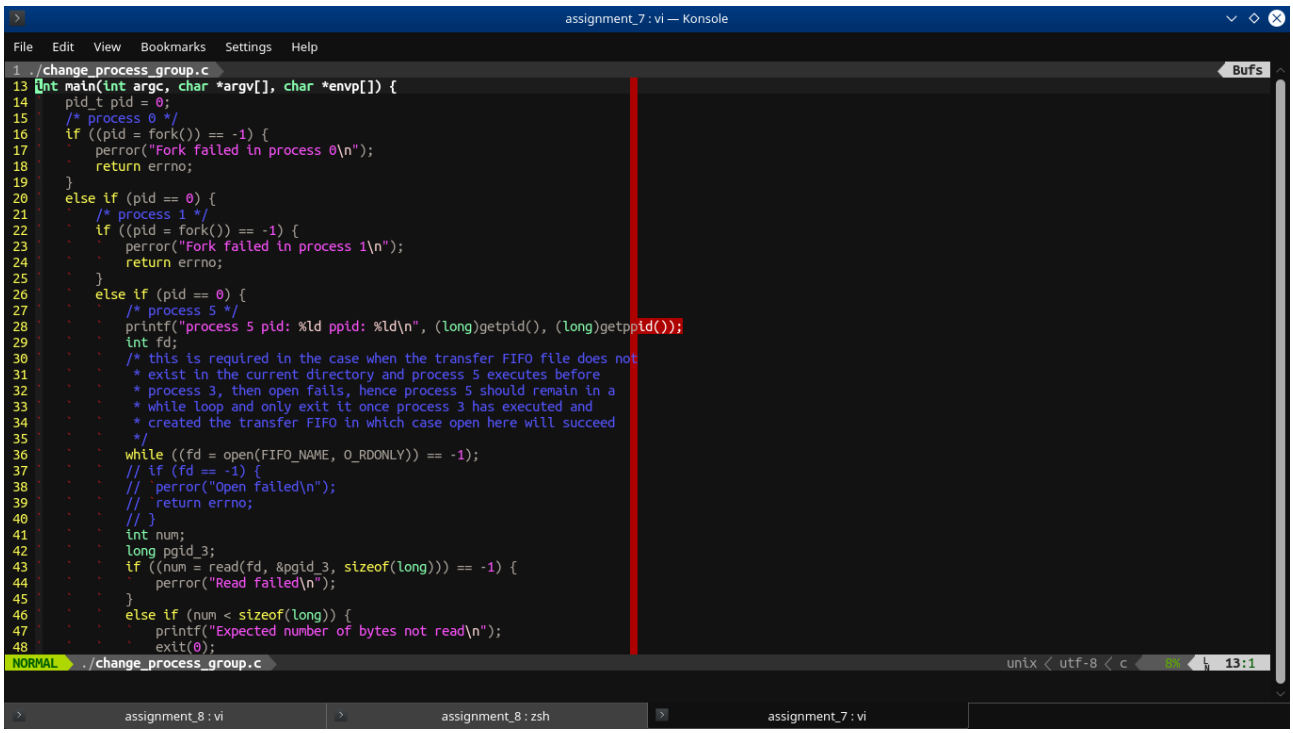
**Output-**


**2.** You have to create a process tree as shown below. Then you create a process group of (3, 4, 5) so that later (not to be implemented now) process 0 can send a signal to this group.

FIFO has been used to transfer the process group id created by the process 3 to process 5 in order for process 5 to be able to change its own pgid to this new pgid value obtained via FIFO transfer. Print statements display the changed pgids of processes 3,4 and 5, before that they also show the pids and ppids of all created processes to determine the correction formation of the process tree. Finally, there are print statements to check proper working of the FIFO.

The below code has been changed from the actual submission to handle the said case in the comment as seen in the output image below. It has been done to handle the case where the transfer FIFO file does not exist in the current directory and process 5 executes before process 3, then open fails, hence, process 5 should remain in a while loop and only exit it once process 3 has executed and created the transfer FIFO in which case open will succeed.

**Output-**


assignment_7 : vi — Konsole
File Edit View Bookmarks Settings Help
1 ./change_process_group.c
Bufs
13 int main(int argc, char *argv[], char *envp[]) {
14 pid_t pid = 0;
15 /* process 0 */
16 if ((pid = fork()) == -1) {
17 perror("Fork failed in process 0\n");
18 return errno;
19 }
20 else if (pid == 0) {
21 /* process 1 */
22 if ((pid = fork()) == -1) {
23 perror("Fork failed in process 1\n");
24 return errno;
25 }
26 else if (pid == 0) {
27 /* process 5 */
28 printf("process 5 pid: %ld ppid: %ld\n", (long)getpid(), (long)getppid());
29 int fd;
30 /* this is required in the case when the transfer FIFO file does not
31 * exist in the current directory and process 5 executes before
32 * process 3, then open fails, hence process 5 should remain in a
33 * while loop and only exit it once process 3 has executed and
34 * created the transfer FIFO in which case open here will succeed
35 */
36 while ((fd = open(FIFO_NAME, O_RDONLY)) == -1);
37 // if (fd == -1) {
38 // perror("Open failed\n");
39 // return errno;
40 // }
41 int num;
42 long pgid_3;
43 if ((num = read(fd, &pgid_3, sizeof(long))) == -1) {
44 perror("Read failed\n");
45 }
46 else if (num < sizeof(long)) {
47 printf("Expected number of bytes not read\n");
48 exit(0);
NORMAL ./change_process_group.c
unix utf-8 c 8% 13:1
assignment_8 : vi
assignment_8 : zsh
assignment_7 : vi