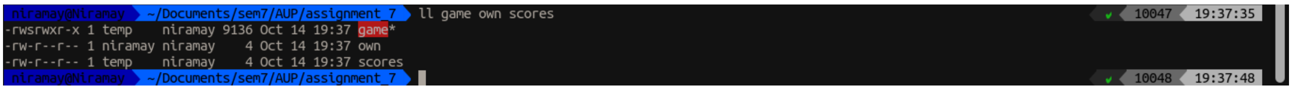
Assignment 7

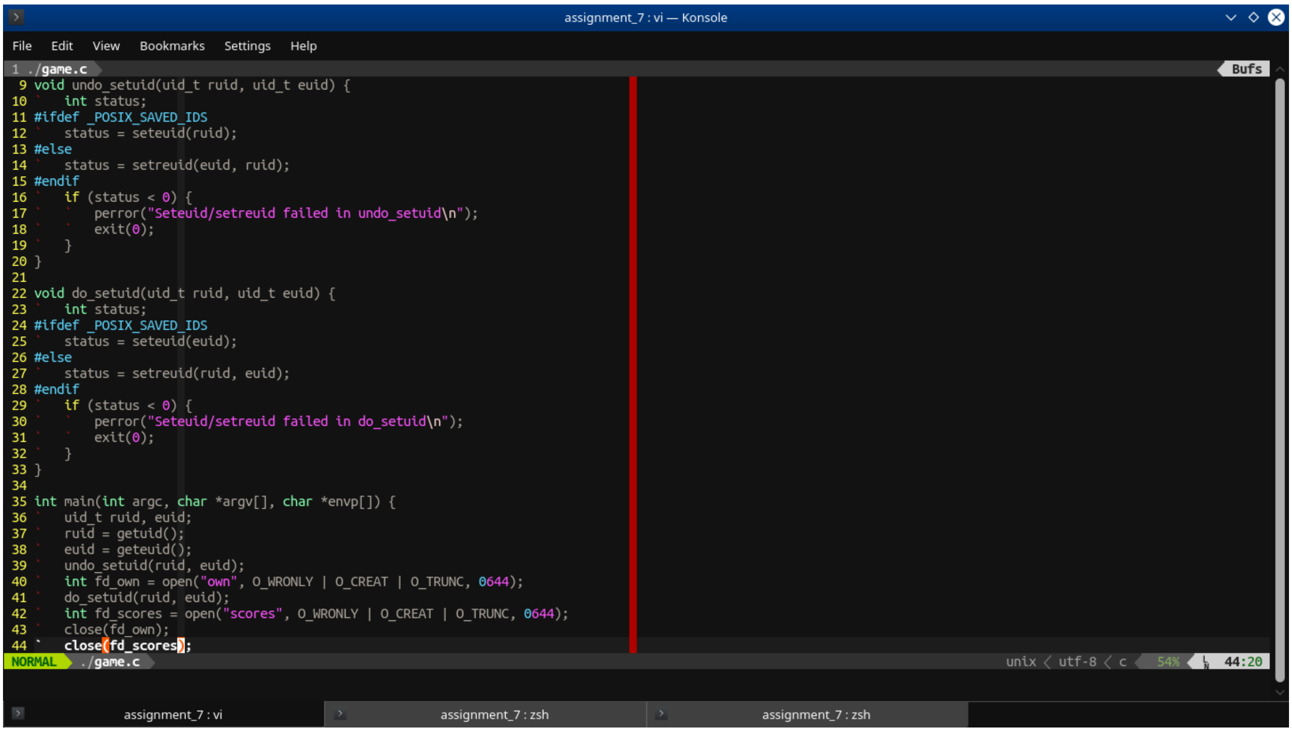
Niramay Vaidya 111605075
Srishti Shelke   111603056

2. 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.

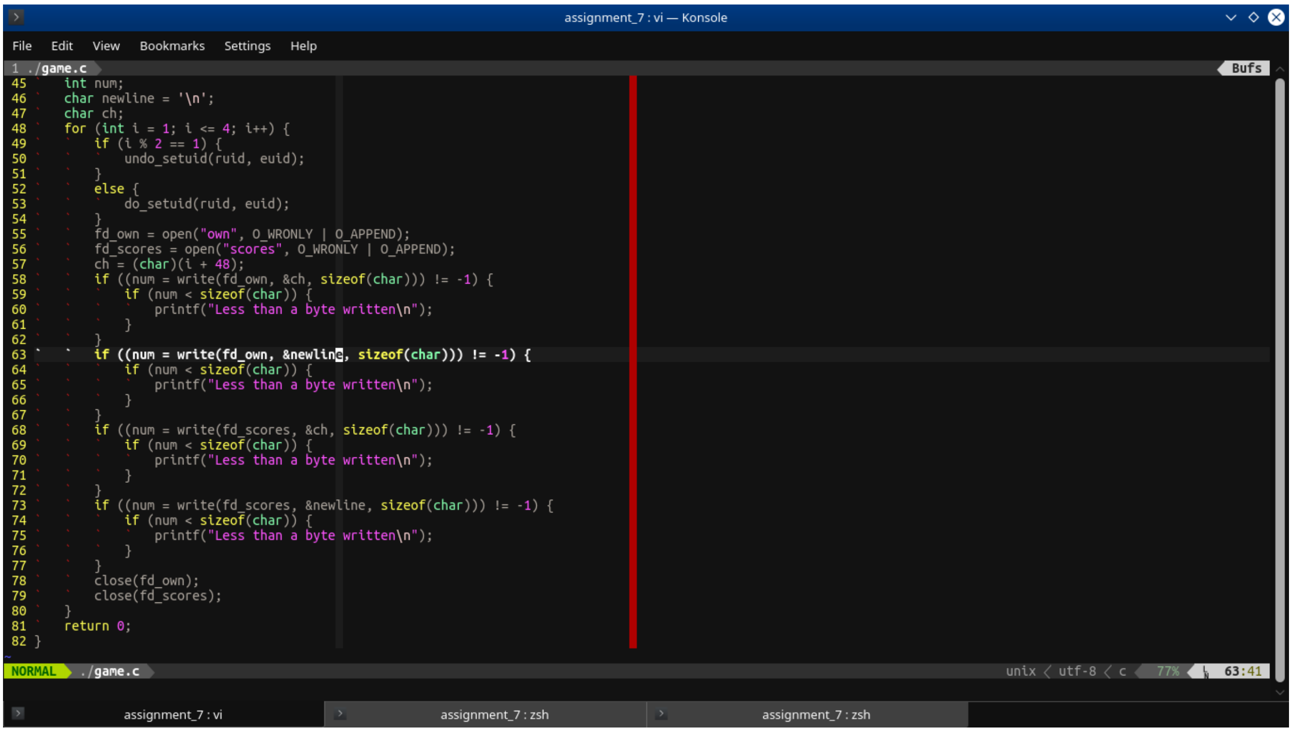


```
niramay@Niramay  ~/Documents/sem7/AUP/assignment_7  ll game own scores          ✔  10047  19:37:35
-rwsrwxr-x 1 temp    niramay 9136 Oct 14 19:37 game*
-rw-r--r-- 1 niramay niramay    4 Oct 14 19:37 own
-rw-r--r-- 1 temp    niramay    4 Oct 14 19:37 scores
niramay@Niramay  ~/Documents/sem7/AUP/assignment_7                               ✔  10048  19:37:48
```

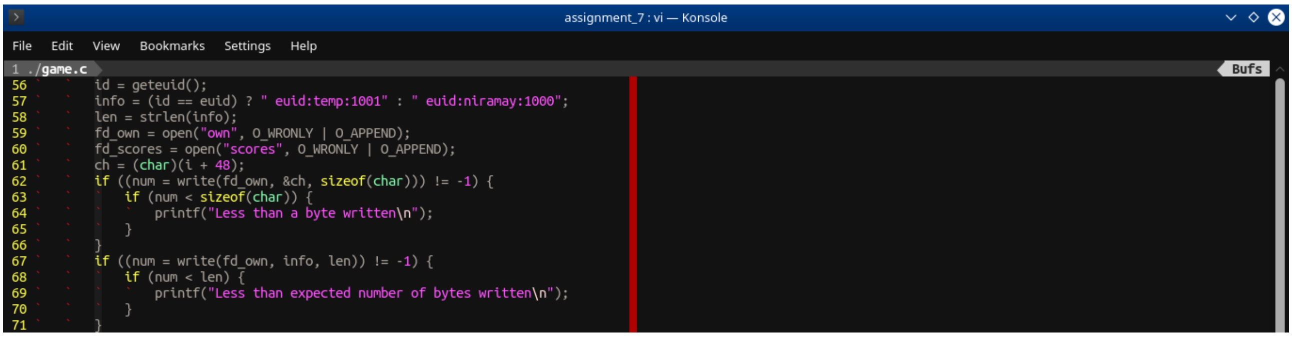
Code-



```c
void undo_setuid(uid_t ruid, uid_t euid) {
    int status;
#ifdef _POSIX_SAVED_IDS
    status = seteuid(ruid);
#else
    status = setreuid(euid, ruid);
#endif
    if (status < 0) {
        perror("Seteuid/setreuid failed in undo_setuid\n");
        exit(0);
    }
}

void do_setuid(uid_t ruid, uid_t euid) {
    int status;
#ifdef _POSIX_SAVED_IDS
    status = seteuid(euid);
#else
    status = setreuid(ruid, euid);
#endif
    if (status < 0) {
        perror("Seteuid/setreuid failed in do_setuid\n");
        exit(0);
    }
}

int main(int argc, char *argv[], char *envp[]) {
    uid_t ruid, euid;
    ruid = getuid();
    euid = geteuid();
    undo_setuid(ruid, euid);
    int fd_own = open("own", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    do_setuid(ruid, euid);
    int fd_scores = open("scores", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    close(fd_own);
    close(fd_scores);
```

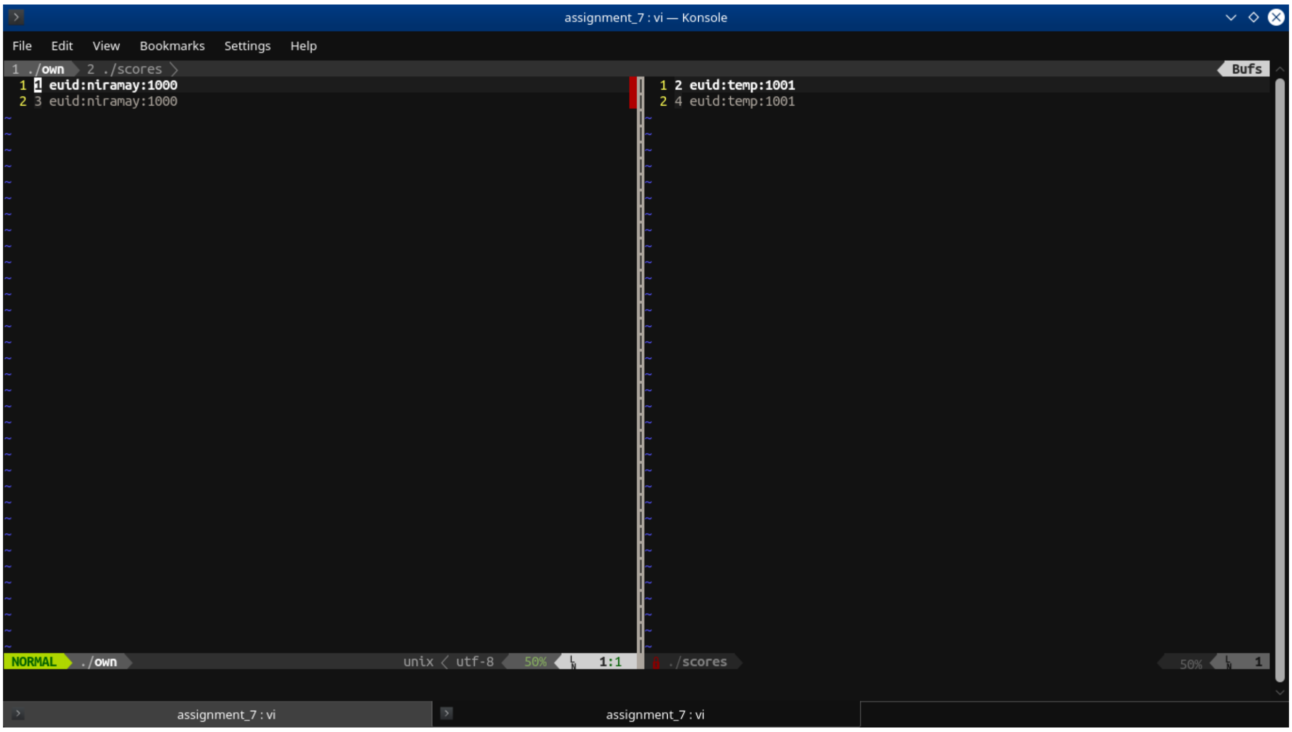

assignment_7 : vi — Konsole
File Edit View Bookmarks Settings Help
1 ./game.c
Bufs
NORMAL ./game.c unix utf-8 c 77% 63:41
assignment_7 : vi
assignment_7 : zsh
assignment_7 : zsh

```c
    int num;
    char newline = '\n';
    char ch;
    for (int i = 1; i <= 4; i++) {
        if (i % 2 == 1) {
            undo_setuid(ruid, euid);
        }
        else {
            do_setuid(ruid, euid);
        }
        fd_own = open("own", O_WRONLY | O_APPEND);
        fd_scores = open("scores", O_WRONLY | O_APPEND);
        ch = (char)(i + 48);
        if ((num = write(fd_own, &ch, sizeof(char))) != -1) {
            if (num < sizeof(char)) {
                printf("Less than a byte written\n");
            }
        }
        if ((num = write(fd_own, &newline, sizeof(char))) != -1) {
            if (num < sizeof(char)) {
                printf("Less than a byte written\n");
            }
        }
        if ((num = write(fd_scores, &ch, sizeof(char))) != -1) {
            if (num < sizeof(char)) {
                printf("Less than a byte written\n");
            }
        }
        if ((num = write(fd_scores, &newline, sizeof(char))) != -1) {
            if (num < sizeof(char)) {
                printf("Less than a byte written\n");
            }
        }
        close(fd_own);
        close(fd_scores);
    }
    return 0;
}
```


assignment_7 : vi — Konsole
File Edit View Bookmarks Settings Help
1 ./game.c
Bufs

```c
        id = geteuid();
        info = (id == euid) ? " euid:temp:1001" : " euid:niramay:1000";
        len = strlen(info);
        fd_own = open("own", O_WRONLY | O_APPEND);
        fd_scores = open("scores", O_WRONLY | O_APPEND);
        ch = (char)(i + 48);
        if ((num = write(fd_own, &ch, sizeof(char))) != -1) {
            if (num < sizeof(char)) {
                printf("Less than a byte written\n");
            }
        }
        if ((num = write(fd_own, info, len)) != -1) {
            if (num < len) {
                printf("Less than expected number of bytes written\n");
            }
        }
```

Output-

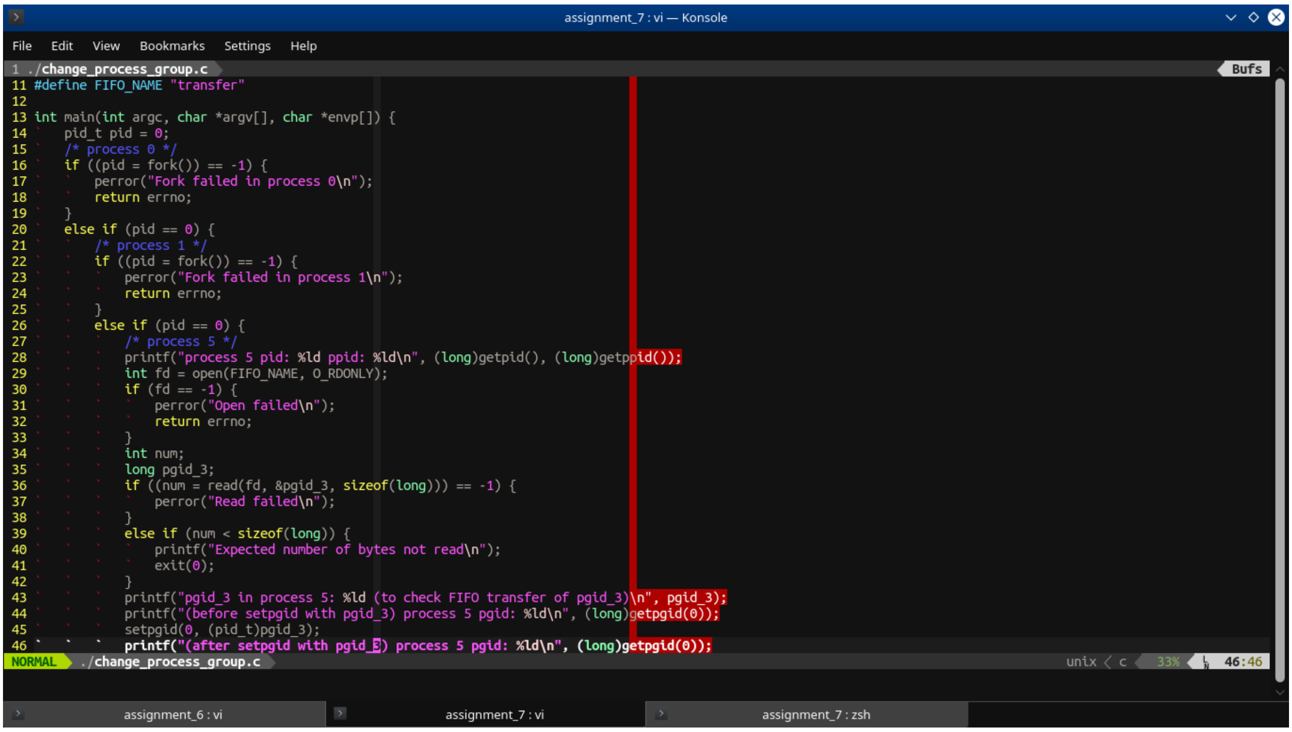

assignment_7 : vi — Konsole
File Edit View Bookmarks Settings Help
1 ./own 2 ./scores
Bufs
NORMAL ./own unix utf-8 50% 1:1
./scores 50% 1
assignment_7 : vi
assignment_7 : vi

```
1 euid:niramay:1000
3 euid:niramay:1000
```

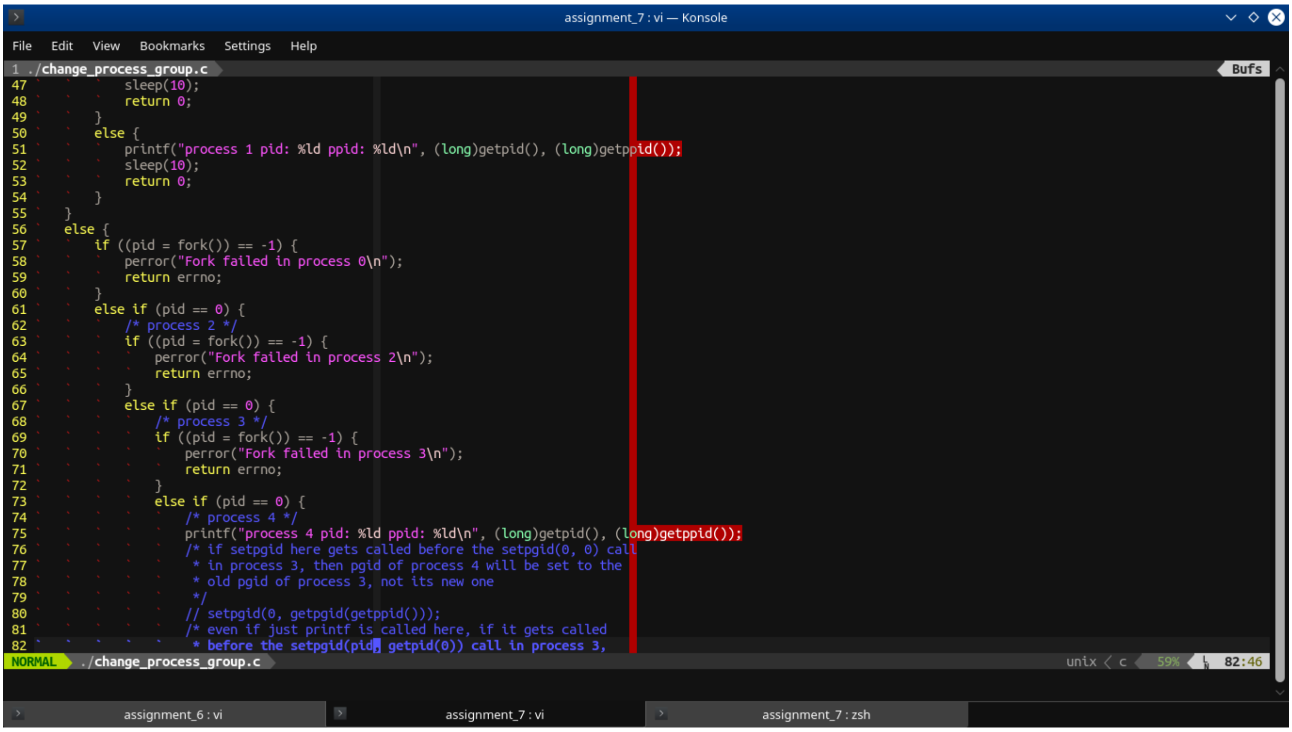
```
2 euid:temp:1001
4 euid:temp:1001
```

1. 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.

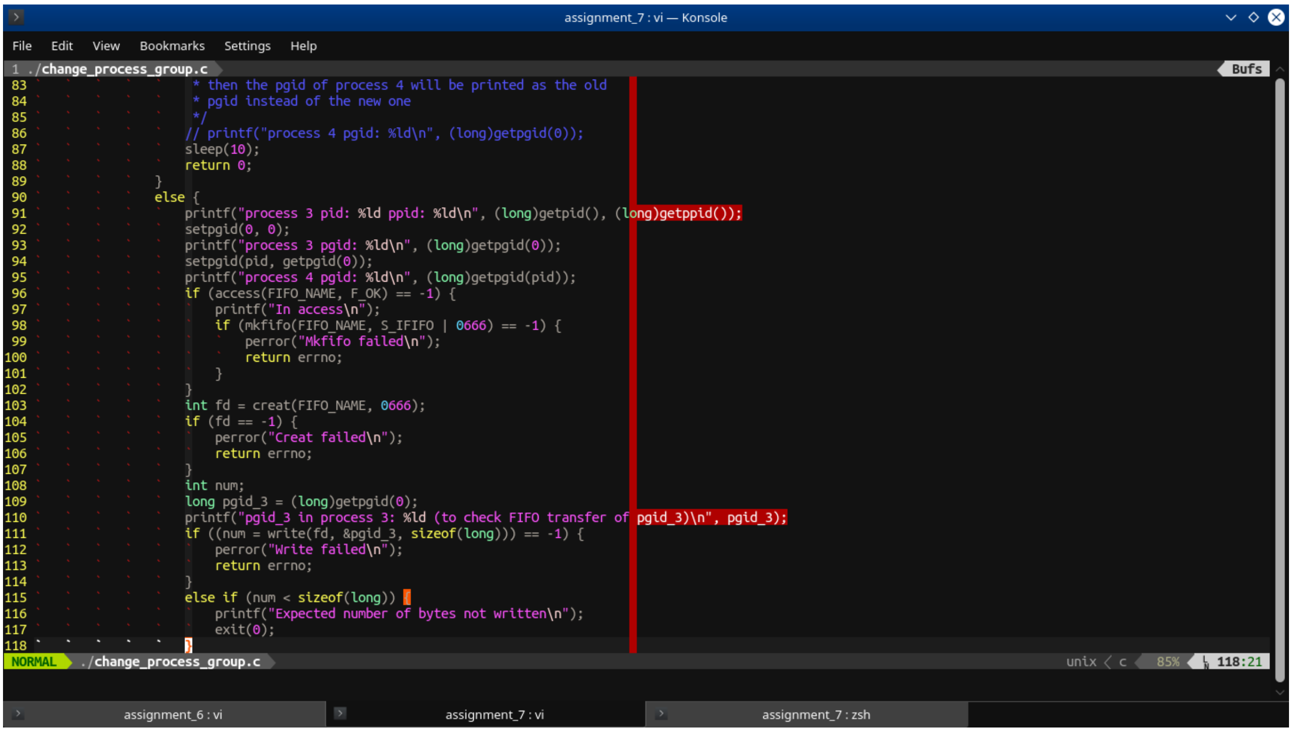
Code-



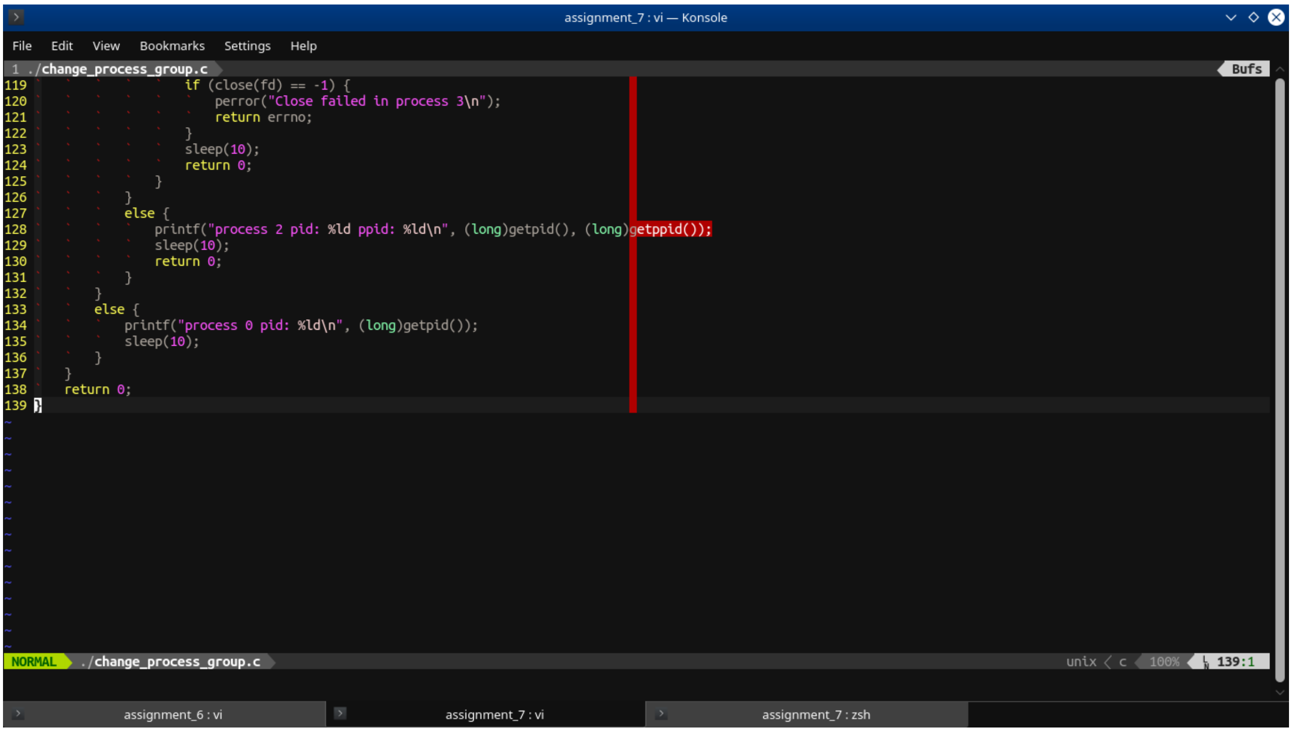
```c
#define FIFO_NAME "transfer"

int main(int argc, char *argv[], char *envp[]) {
    pid_t pid = 0;
    /* process 0 */
    if ((pid = fork()) == -1) {
        perror("Fork failed in process 0\n");
        return errno;
    }
    else if (pid == 0) {
        /* process 1 */
        if ((pid = fork()) == -1) {
            perror("Fork failed in process 1\n");
            return errno;
        }
        else if (pid == 0) {
            /* process 5 */
            printf("process 5 pid: %ld ppid: %ld\n", (long)getpid(), (long)getppid());
            int fd = open(FIFO_NAME, O_RDONLY);
            if (fd == -1) {
                perror("Open failed\n");
                return errno;
            }
            int num;
            long pgid_3;
            if ((num = read(fd, &pgid_3, sizeof(long))) == -1) {
                perror("Read failed\n");
            }
            else if (num < sizeof(long)) {
                printf("Expected number of bytes not read\n");
                exit(0);
            }
            printf("pgid_3 in process 5: %ld (to check FIFO transfer of pgid_3)\n", pgid_3);
            printf("(before setpgid with pgid_3) process 5 pgid: %ld\n", (long)getpgid(0));
            setpgid(0, (pid_t)pgid_3);
            printf("(after setpgid with pgid_3) process 5 pgid: %ld\n", (long)getpgid(0));
```



```c
            sleep(10);
            return 0;
        }
        else {
            printf("process 1 pid: %ld ppid: %ld\n", (long)getpid(), (long)getppid());
            sleep(10);
            return 0;
        }
    }
    else {
        if ((pid = fork()) == -1) {
            perror("Fork failed in process 0\n");
            return errno;
        }
        else if (pid == 0) {
            /* process 2 */
            if ((pid = fork()) == -1) {
                perror("Fork failed in process 2\n");
                return errno;
            }
            else if (pid == 0) {
                /* process 3 */
                if ((pid = fork()) == -1) {
                    perror("Fork failed in process 3\n");
                    return errno;
                }
                else if (pid == 0) {
                    /* process 4 */
                    printf("process 4 pid: %ld ppid: %ld\n", (long)getpid(), (long)getppid());
                    /* if setpgid here gets called before the setpgid(0, 0) call
                     * in process 3, then pgid of process 4 will be set to the
                     * old pgid of process 3, not its new one
                     */
                    // setpgid(0, getpgid(getppid()));
                    /* even if just printf is called here, if it gets called
                     * before the setpgid(pid, getpid(0)) call in process 3,
```


assignment_7 : vi — Konsole
File Edit View Bookmarks Settings Help
1 ./change_process_group.c
Bufs
NORMAL ./change_process_group.c
unix c 85% 118:21
assignment_6 : vi
assignment_7 : vi
assignment_7 : zsh

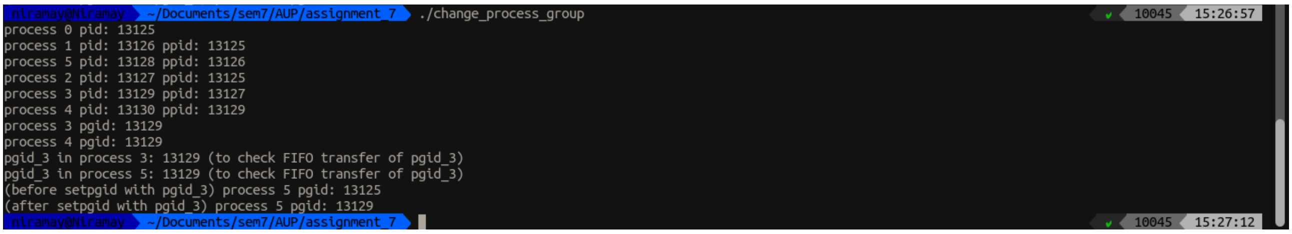
```
                     * then the pgid of process 4 will be printed as the old
                     * pgid instead of the new one
                     */
                    // printf("process 4 pgid: %ld\n", (long)getpgid(0));
                    sleep(10);
                    return 0;
                }
                else {
                    printf("process 3 pid: %ld ppid: %ld\n", (long)getpid(), (long)getppid());
                    setpgid(0, 0);
                    printf("process 3 pgid: %ld\n", (long)getpgid(0));
                    setpgid(pid, getpgid(0));
                    printf("process 4 pgid: %ld\n", (long)getpgid(pid));
                    if (access(FIFO_NAME, F_OK) == -1) {
                        printf("In access\n");
                        if (mkfifo(FIFO_NAME, S_IFIFO | 0666) == -1) {
                            perror("Mkfifo failed\n");
                            return errno;
                        }
                    }
                    int fd = creat(FIFO_NAME, 0666);
                    if (fd == -1) {
                        perror("Creat failed\n");
                        return errno;
                    }
                    int num;
                    long pgid_3 = (long)getpgid(0);
                    printf("pgid_3 in process 3: %ld (to check FIFO transfer of pgid_3)\n", pgid_3);
                    if ((num = write(fd, &pgid_3, sizeof(long))) == -1) {
                        perror("Write failed\n");
                        return errno;
                    }
                    else if (num < sizeof(long)) {
                        printf("Expected number of bytes not written\n");
                        exit(0);
                    }
```


assignment_7 : vi — Konsole
File Edit View Bookmarks Settings Help
1 ./change_process_group.c
Bufs
NORMAL ./change_process_group.c
unix c 100% 139:1
assignment_6 : vi
assignment_7 : vi
assignment_7 : zsh

```
                    if (close(fd) == -1) {
                        perror("Close failed in process 3\n");
                        return errno;
                    }
                    sleep(10);
                    return 0;
                }
            }
            else {
                printf("process 2 pid: %ld ppid: %ld\n", (long)getpid(), (long)getppid());
                sleep(10);
                return 0;
            }
        }
        else {
            printf("process 0 pid: %ld\n", (long)getpid());
            sleep(10);
        }
    }
    return 0;
}
```

(error handling has been done for setpgid() later on but is not visible in the screenshots of the code)

Output-



```
niramay@Niramay ~/Documents/sem7/AUP/assignment_7 ./change_process_group          10045 15:26:57
process 0 pid: 13125
process 1 pid: 13126 ppid: 13125
process 5 pid: 13128 ppid: 13126
process 2 pid: 13127 ppid: 13125
process 3 pid: 13129 ppid: 13127
process 4 pid: 13130 ppid: 13129
process 3 pgid: 13129
process 4 pgid: 13129
pgid_3 in process 3: 13129 (to check FIFO transfer of pgid_3)
pgid_3 in process 5: 13129 (to check FIFO transfer of pgid_3)
(before setpgid with pgid_3) process 5 pgid: 13125
(after setpgid with pgid_3) process 5 pgid: 13129
niramay@Niramay ~/Documents/sem7/AUP/assignment_7                                 10045 15:27:12
```