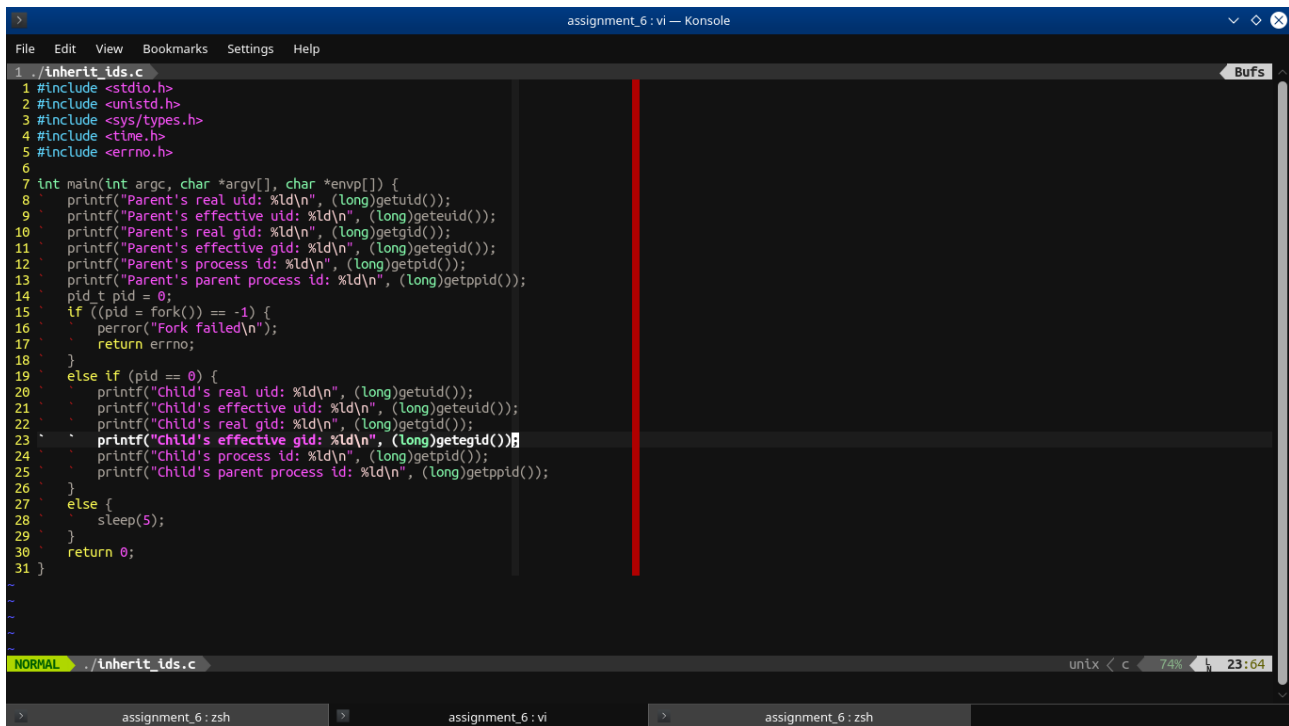


Assignment 6

Srishti Shelke 111603056
Niramay Vaidya 111605075

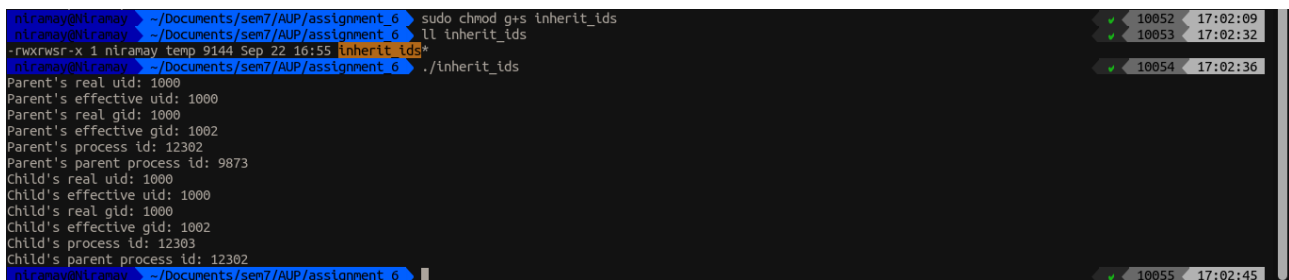
1. A child process inherits real user id, real group id, effective user id and effective group id of the parent process, while process id and parent process id are not. Demonstrate.

Code-



```
1 ./inherit_ids.c
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <time.h>
6 #include <errno.h>
7
8 int main(int argc, char *argv[], char *envp[]) {
9     printf("Parent's real uid: %ld\n", (long)getuid());
10    printf("Parent's effective uid: %ld\n", (long)geteuid());
11    printf("Parent's real gid: %ld\n", (long)getgid());
12    printf("Parent's effective gid: %ld\n", (long)getegid());
13    printf("Parent's process id: %ld\n", (long)getpid());
14    printf("Parent's parent process id: %ld\n", (long)getppid());
15    pid_t pid = 0;
16    if ((pid = fork()) == -1) {
17        perror("Fork failed\n");
18        return errno;
19    }
20    else if (pid == 0) {
21        printf("Child's real uid: %ld\n", (long)getuid());
22        printf("Child's effective uid: %ld\n", (long)geteuid());
23        printf("Child's real gid: %ld\n", (long)getgid());
24        printf("Child's effective gid: %ld\n", (long)getegid());
25        printf("Child's process id: %ld\n", (long)getpid());
26        printf("Child's parent process id: %ld\n", (long)getppid());
27    }
28    else {
29        sleep(5);
30    }
31    return 0;
32 }
```

Output-



```
niranay@niranay: ~/Documents/sem7/AUP/assignment_6 $ sudo chmod g+s inherit_ids
niranay@niranay: ~/Documents/sem7/AUP/assignment_6 $ ll inherit_ids
-rwxrwxr-x 1 niranay temp 9144 Sep 22 16:55 inherit_ids
niranay@niranay: ~/Documents/sem7/AUP/assignment_6 $ ./inherit_ids
Parent's real uid: 1000
Parent's effective uid: 1000
Parent's real gid: 1000
Parent's effective gid: 1002
Parent's process id: 12302
Parent's parent process id: 9873
Child's real uid: 1000
Child's effective uid: 1000
Child's real gid: 1000
Child's effective gid: 1002
Child's process id: 12303
Child's parent process id: 12302
niranay@niranay: ~/Documents/sem7/AUP/assignment_6 $
```

2. Verify whether it is possible for a child process to handle a file opened by its parent Immediately after the fork() call?

Code-

```
1 ./child_file_handle.c
2 #include <stdio.h>
3 #include <sys/types.h>
4 #include <sys/stat.h>
5 #include <fcntl.h>
6 #include <unistd.h>
7 #include <errno.h>
8 #include <time.h>
9 #include <stdlib.h>
10 int main(int argc, char *argv[], char *envp[]) {
11     int fd = open("parent_file", O_RDWR | O_CREAT | O_APPEND, 0666);
12     pid_t pid = 0;
13     if ((pid = fork()) == -1) {
14         perror("Fork failed\n");
15         return errno;
16     }
17     else if (pid == 0) {
18         int size = 0;
19         if ((size = write(fd, "coep", sizeof(char) * 4)) == -1) {
20             perror("Write failed\n");
21             return errno;
22         }
23         else if (size < (sizeof(char) * 4)) {
24             printf("Less number of bytes written than provided\n");
25             exit(0);
26         }
27     }
28     else {
29         sleep(5);
30     }
31     return 0;
32 }
```

Output-

```
niramay@niramay: ~/Documents/sem7/AUP/assignment_6
niramay@niramay: ~/Documents/sem7/AUP/assignment_6 > ./child_file_handle
-rw-rw-r-- 1 niramay niramay 4 Sep 22 23:20 parent_file
niramay@niramay: ~/Documents/sem7/AUP/assignment_6 > ./child_file_handle
-rw-rw-r-- 1 niramay niramay 8 Sep 22 23:21 parent_file
niramay@niramay: ~/Documents/sem7/AUP/assignment_6 >
```

```
1 ./parent_file
2 #include <stdio.h>
3 #include <sys/types.h>
4 #include <sys/stat.h>
5 #include <fcntl.h>
6 #include <unistd.h>
7 #include <errno.h>
8 #include <time.h>
9 #include <stdlib.h>
10 int main(int argc, char *argv[], char *envp[]) {
11     int fd = open("parent_file", O_RDWR | O_CREAT | O_APPEND, 0666);
12     pid_t pid = 0;
13     if ((pid = fork()) == -1) {
14         perror("Fork failed\n");
15         return errno;
16     }
17     else if (pid == 0) {
18         int size = 0;
19         if ((size = write(fd, "coep", sizeof(char) * 4)) == -1) {
20             perror("Write failed\n");
21             return errno;
22         }
23         else if (size < (sizeof(char) * 4)) {
24             printf("Less number of bytes written than provided\n");
25             exit(0);
26         }
27     }
28     else {
29         sleep(5);
30     }
31     return 0;
32 }
```

3. Measures the performance of the getpid() and the fork functions using gettimeofday to measure the execution time. Measure the performance ten times for each of the two system calls and provide the timing results and compute an average for each system call.

Code-

```
assignment_6: vi — Konsole
File Edit View Bookmarks Settings Help

1 //execution_time.c
7
8 int main(int argc, char *argv[], char *envp[]) {
9     pid_t pid = 0;
10    float getpid_t = 0;
11    float fork_t = 0;
12    struct timeval tv;
13    suseconds_t initial_t = 0;
14    printf("Timings results are in microseconds\n");
15    for (int i = 0; i < 10; i++) {
16        if (gettimeofday(&tv, NULL) == -1) {
17            perror("Gettimeofday failed\n");
18            return errno;
19        }
20        initial_t = tv.tv_usec;
21        pid = getpid();
22        if (gettimeofday(&tv, NULL) == -1) {
23            perror("Gettimeofday failed\n");
24            return errno;
25        }
26        printf("getpid (%d) : %d\n", i + 1, (int)(tv.tv_usec - initial_t));
27        getpid_t += (float)(tv.tv_usec - initial_t);
28        if (gettimeofday(&tv, NULL) == -1) {
29            perror("Gettimeofday failed\n");
30            return errno;
31        }
32        initial_t = tv.tv_usec;
33        if ((pid = fork()) == -1) {
34            perror("Fork failed\n");
35            return errno;
36        }
37        else if (pid == 0) {
38            exit(1);
39        }
40        if (gettimeofday(&tv, NULL) == -1) {
41            perror("Gettimeofday failed\n");
42            return errno;
43        }
44        printf("fork (%d) : %d\n", i + 1, (int)(tv.tv_usec - initial_t));
45        fork_t += (float)(tv.tv_usec - initial_t);
46    }
47    getpid_t /= 10.0;
48    fork_t /= 10.0;
49    printf("getpid avg : %f\n", getpid_t);
50    printf("fork avg : %f\n", fork_t);
51    return 0;
52 }
```

Output-

```
nik@nikanov:~/Documents/sem7/AUP/assignment_6$ ./execution_time
Timings results are in microseconds
getpid (1) : 19
fork (1) : 172
getpid (2) : 2
fork (2) : 114
getpid (3) : 1
fork (3) : 175
getpid (4) : 2
fork (4) : 207
getpid (5) : 1
fork (5) : 159
getpid (6) : 2
fork (6) : 182
getpid (7) : 2
fork (7) : 131
getpid (8) : 2
fork (8) : 118
getpid (9) : 1
fork (9) : 140
getpid (10) : 1
fork (10) : 120
getpid avg : 3.300000
fork avg : 151.800003
nik@nikanov:~/Documents/sem7/AUP/assignment_6$
```