Advanced Unix Programming
Lab 5

Purva Tendulkar : 111403049
Shruti Dogra : 111403075

Q1. 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.

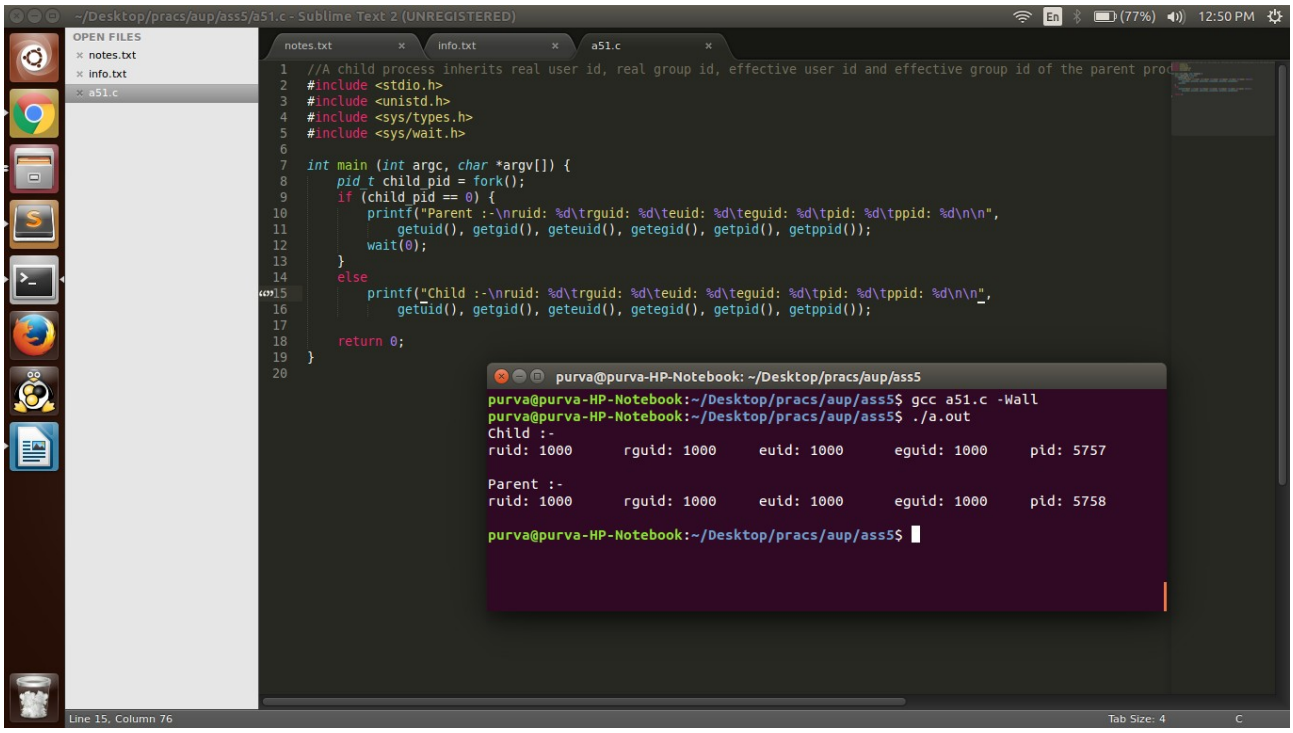
Code :

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

int main (int argc, char *argv[]) {
	pid_t child_pid = fork();
	if (child_pid == 0) {
		printf("Parent :-\nruid: %d\trguid: %d\teuid: %d\teguid: %d\tpid: %d\tppid: %d\n\n",
			getuid(), getgid(), geteuid(), getegid(), getpid(), getppid());
		wait(0);
	}
	else
		printf("Child :-\nruid: %d\trguid: %d\teuid: %d\teguid: %d\tpid: %d\tppid: %d\n\n",
			getuid(), getgid(), geteuid(), getegid(), getpid(), getppid());

	return 0;
}
```

Input & Output Screenshots :



Q2. Verify whether it is possible for a child process to handle a file opened by its parent Immediately after the fork() call?

Code :

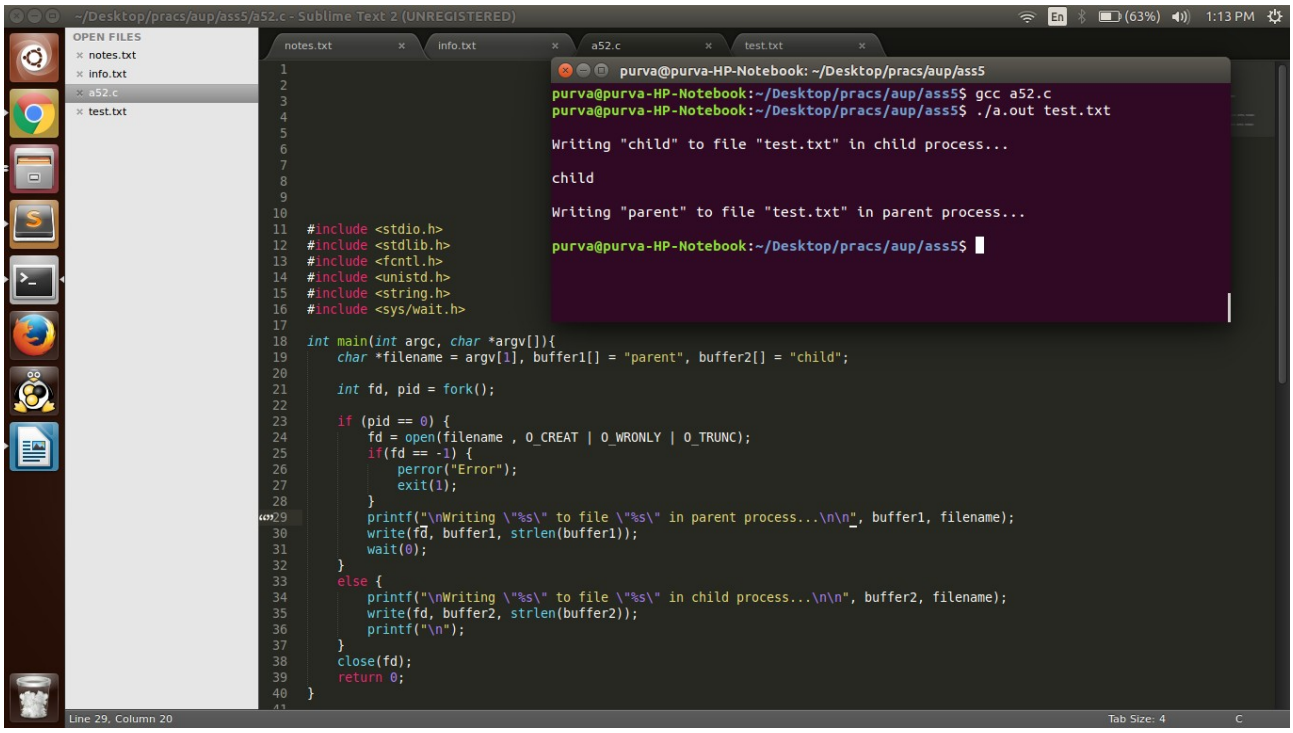
```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

int main(int argc, char *argv[]){
        char *filename = argv[1], buffer1[] = "parent", buffer2[] = "child";

        int fd, pid = fork();

        if (pid == 0) {
                fd = open(filename , O_CREAT | O_WRONLY | O_TRUNC);
                if(fd == -1) {
                        perror("Error ");
                        exit(1);
                }
                printf("Writing \"%s\" to file \"%s\" in parent process...\n", buffer1, filename);
                write(fd, buffer1, strlen(buffer1));
                wait(0);
        }
        else {
                sleep(2);
                printf("Writing \"%s\" to file \"%s\" in child process...\n", buffer2, filename);
                write(fd, buffer2, strlen(buffer2));
        }
        close(fd);
        return 0;
}
```

Input and Output Screenshots :



Explanation :

1. If you open a file in the parent process after fork() it won't be shared with the child.
2. “child” is printed to the terminal (stdout) because the fd in the child assumes value 1 and not the fd in the parent process.

Q3. The parent starts as many child processes as to the value of its integer command line argument. The child processes simply sleep for the time specified by the argument, then exit. After starting all the children, the parent process must wait until they have all terminated before terminating itself.

Code :

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

int main(int argc, char *argv[]){
	int pid, sleep_time, process, i;
	process = atoi(argv[1]);
	sleep_time = atoi(argv[2]);

	printf("Creating %d children.\n", process);
	for (i = 0; i < process; i++){
		pid = fork();
		if (pid == 0) {
			printf("Sleeping\n");
			sleep(sleep_time);
			exit(0);
		}
		else if (pid != -1) {
			printf("pid : %d \n", pid);
			waitpid(pid - 1, NULL, 0);
		}
		else
			printf("Error in fork \n");
	}
	wait(0);
	printf("All processes exited. \n");

	return 0;
}
```

Input & Output Screenshots :

~/Desktop/pracs/aup/ass5/a53.c - Sublime Text 2 (UNREGISTERED)

OPEN FILES
- notes.txt
- info.txt
- a53.c

notes.txt | info.txt | a53.c

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

int main(int argc, char *argv[]){
    int pid, sleep_time, process, i;
    process = atoi(argv[1]);
    sleep_time = atoi(argv[2]);

    printf("Creating %d children.\n", process);
    for (i = 0; i < process; i++){
        pid = fork();
        if (pid == 0) {
            printf("Sleeping\n");
            sleep(sleep_time);
            exit(0);
        }
        else if (pid != -1) {
            printf("pid : %d \n", pid);
            waitpid(pid - 1, NULL, 0);
        }
        else
            printf("Error in fork \n");
    }
    wait(0);
    printf("All processes exited. \n");

    return 0;
}
```

purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass5

```
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass5$ gcc a53.c -Wall
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass5$ ./a.out 7 4
Creating 7 children.
pid : 7415
Sleeping
pid : 7416
Sleeping
pid : 7432
Sleeping
pid : 7433
Sleeping
pid : 7434
Sleeping
pid : 7435
Sleeping
pid : 7438
All processes exited.
Sleeping
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass5$
```

Line 23, Column 10 | Tab Size: 4 | C