

OS Lab 3

Name: Paawan Kohli
Reg No: 180905416
Roll No: 52

Q1. Write a C program to block a parent process until the child completes using a wait system call.

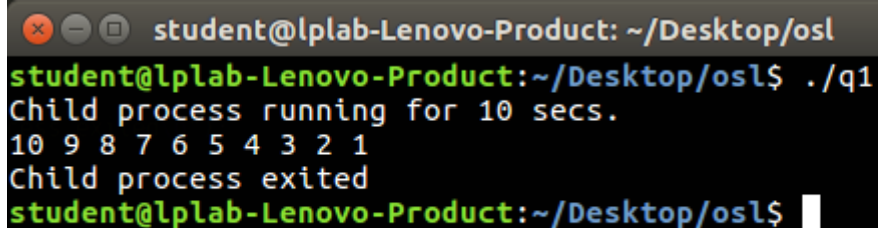
```
#include <unistd.h> // unix std lib
#include <sys/types.h> // for pid_t
#include <sys/wait.h> // for wait(int*)
#include <stdio.h>
#include <stdlib.h>

void main () {
    pid_t pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
        exit(-1);
    } else if (pid == 0) {
        int secs = 10;
        printf("Child process running for %d secs.\n", secs);

        for (int i = secs; i > 0 ; i--) {
            printf("%d ", i); fflush(stdout);
            sleep(1);
        }

        printf("\n");
        exit(0);
    } else {
        wait(NULL);
        printf("Child process exited\n");
        exit(0);
    }
}
```



A terminal window titled 'student@lplab-Lenovo-Product: ~/Desktop/osl' shows the execution of the program. The command './q1' is entered, and the output is: 'Child process running for 10 secs.', followed by a countdown '10 9 8 7 6 5 4 3 2 1', then 'Child process exited', and finally the prompt 'student@lplab-Lenovo-Product:~/Desktop/osl\$'.

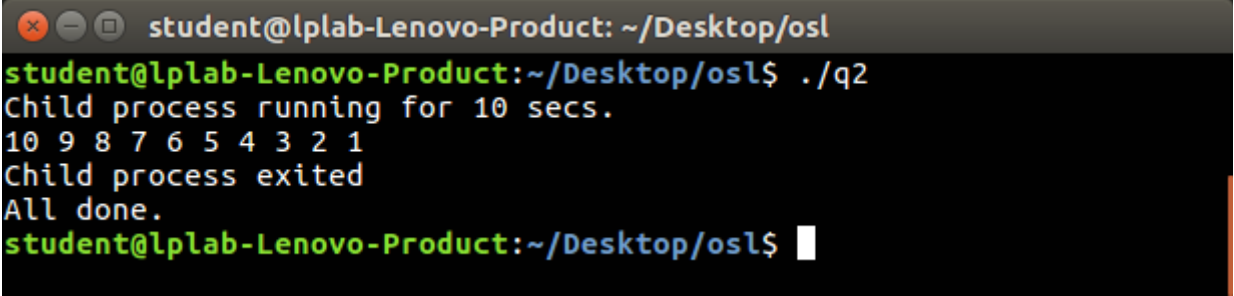
```
student@lplab-Lenovo-Product: ~/Desktop/osl
student@lplab-Lenovo-Product:~/Desktop/osl$ ./q1
Child process running for 10 secs.
10 9 8 7 6 5 4 3 2 1
Child process exited
student@lplab-Lenovo-Product:~/Desktop/osl$
```

Q2. Write a C program to load the binary executable of the previous program in a child process using the exec system call.

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

void main() {
    pid_t pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
        exit(-1);
    } else if (pid == 0) {
        execvp("./q1", NULL);
        exit(0);
    } else {
        wait(NULL);
        printf("All done.\n");
        exit(0);
    }
}
```



A terminal window titled 'student@lplab-Lenovo-Product: ~/Desktop/osl' shows the execution of a C program. The user enters './q2' at the prompt. The program outputs 'Child process running for 10 secs.', followed by a countdown '10 9 8 7 6 5 4 3 2 1', then 'Child process exited', and finally 'All done.'. The prompt returns to 'student@lplab-Lenovo-Product: ~/Desktop/osl\$'.

```
student@lplab-Lenovo-Product: ~/Desktop/osl
student@lplab-Lenovo-Product:~/Desktop/osl$ ./q2
Child process running for 10 secs.
10 9 8 7 6 5 4 3 2 1
Child process exited
All done.
student@lplab-Lenovo-Product:~/Desktop/osl$
```

Q3. Write a program to create a child process. Display the process IDs of the process, parent and child (if any) in both parent and child processes.

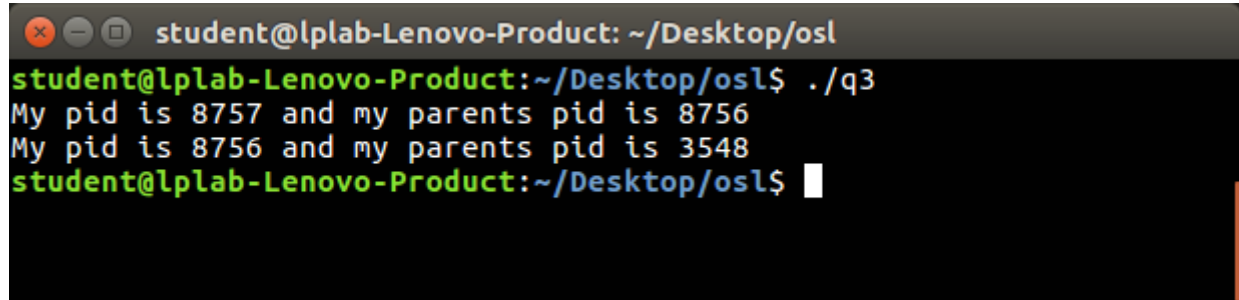
```
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>

void main() {

    pid_t pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
        exit(-1);
    }

    // no need of if condition as only the parent will wait
    // the child will call printf before the parent always
    wait(NULL);
    printf("My pid is %d and my parents pid is %d\n", getpid(), getppid());
    exit(0);
}
```



A terminal window titled 'student@lplab-Lenovo-Product: ~/Desktop/osl' shows the execution of the program. The user runs './q3' and the output is displayed in two lines: 'My pid is 8757 and my parents pid is 8756' and 'My pid is 8756 and my parents pid is 3548'. The prompt returns to 'student@lplab-Lenovo-Product:~/Desktop/osl\$'.

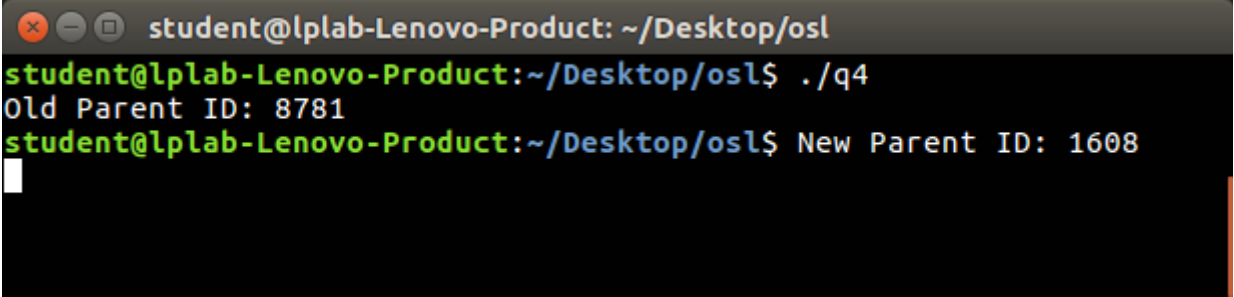
```
student@lplab-Lenovo-Product: ~/Desktop/osl
student@lplab-Lenovo-Product:~/Desktop/osl$ ./q3
My pid is 8757 and my parents pid is 8756
My pid is 8756 and my parents pid is 3548
student@lplab-Lenovo-Product:~/Desktop/osl$
```

Q4. Create a orphan child process and allow init process to adopt it (after parent terminates).

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

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
        exit(-1);
    } else if (pid == 0) {
        printf("Old Parent ID: %d\n", getppid());
        sleep(2);
        printf("New Parent ID: %d\n", getppid());
        exit(0);
    } else {
        sleep(1);
    }
}
```



```
student@lplab-Lenovo-Product: ~/Desktop/osl
student@lplab-Lenovo-Product:~/Desktop/osl$ ./q4
Old Parent ID: 8781
student@lplab-Lenovo-Product:~/Desktop/osl$ New Parent ID: 1608
```