OS LAB 4

23K-0057

Q1.

```
~/Desktop$ cd lab
~/Desktop/lab$ gedit q1.cpp
~/Desktop/lab$ g++ q1.cpp -o q1
~/Desktop/lab$ ./q1
```

```
Child 1: My PID is 3955
Child 2: My parent's ID(PPID) is 3954
Parent: Both children have executed. Exiting now....
```

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

int main() {
    pid_t child1, child2;

    child1 = fork();
    if (child1 == 0) {
        // First child prints its PID
        std::cout << "Child 1: My PID is " << getpid() << std::endl;
        exit(0);
    }

    child2 = fork();
    if (child2 == 0) {
        // Second child prints its parent's PID
        std::cout << "Child 2: My parent's ID(PPID) is " << getppid() << std::endl;
        exit(0);
    }

    // Parent waits for both children
    wait(NULL);
    wait(NULL);

    std::cout << "Parent: Both children have executed. Exiting now...." << std::endl;
    return 0;
}
```

Q2.

```
~/Desktop/lab$ g++ q2.cpp -o q2
~/Desktop/lab$ ./q2
```

```
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
I am a parent process
```

```
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
I am a child process
```

```cpp
#include <iostream>
#include <unistd.h>

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

    if (pid == 0) {
        for (int i = 0; i < 100; i++) {
            std::cout << "I am a child process" << std::endl;
        }
    } else {
        for (int i = 0; i < 100; i++) {
            std::cout << "I am a parent process" << std::endl;
        }
    }

    return 0;
}
```

Q3.

```
~/Desktop/lab$ g++ q3.cpp -o q3
~/Desktop/lab$ ./q3
```

```
Current Process ID (PID) is: 4227
Parent Process ID (PPID) is: 2851
User ID (UID) is: 1001
```

```cpp
#include <iostream>
#include <unistd.h>
#include <sys/types.h>

int main() {
    std::cout << "Current Process ID (PID) is: " << getpid() << std::endl;
    std::cout << "Parent Process ID (PPID) is: " << getppid() << std::endl;
    std::cout << "User ID (UID) is: " << getuid() << std::endl;
    return 0;
}
```

Q4.

```
~/Desktop/lab$ g++ q4.cpp -o q4
~/Desktop/lab$ ./q4
```

```
Copied successfully!
~/Desktop/lab$ cat output.txt
os lab 4 tasks
```

```cpp
#include <iostream>
#include <fcntl.h>
#include <unistd.h>

#define BUFFER_SIZE 1024

int main() {
    int inputFile = open("input.txt", O_RDONLY);
    if (inputFile < 0) {
        perror("Error opening input file... Try again!");
        return 1;
    }

    int outputFile = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (outputFile < 0) {
        perror("Error opening output file...TRY AGAIN!!");
        close(inputFile);
        return 1;
    }

    char buffer[BUFFER_SIZE];
    ssize_t bytesRead;

    while ((bytesRead = read(inputFile, buffer, BUFFER_SIZE)) > 0) {
        write(outputFile, buffer, bytesRead);
    }

    close(inputFile);
    close(outputFile);

    std::cout << "Copied successfully!" << std::endl;
    return 0;
```

Q5.

```
~/Desktop/lab$ g++ q5.cpp -o q5
~/Desktop/lab$ ./q5
```

```
total 132
-rw-rw-r-- 1 oslab oslab    16 Feb 19 20:10 input.txt
-rw-r--r-- 1 oslab oslab    16 Feb 19 20:10 output.txt
-rwxrwxr-x 1 oslab oslab 16752 Feb 19 20:00 q1
-rw-rw-r-- 1 oslab oslab   655 Feb 19 19:39 q1.c
-rw-rw-r-- 1 oslab oslab   675 Feb 19 20:00 q1.cpp
-rwxrwxr-x 1 oslab oslab 16536 Feb 19 20:04 q2
-rw-rw-r-- 1 oslab oslab   361 Feb 19 20:03 q2.cpp
-rwxrwxr-x 1 oslab oslab 16720 Feb 19 20:07 q3
-rw-rw-r-- 1 oslab oslab   309 Feb 19 20:06 q3.cpp
-rwxrwxr-x 1 oslab oslab 16760 Feb 19 20:10 q4
-rw-rw-r-- 1 oslab oslab   766 Feb 19 20:08 q4.cpp
-rwxrwxr-x 1 oslab oslab 16704 Feb 19 20:13 q5
-rw-rw-r-- 1 oslab oslab   361 Feb 19 20:13 q5.cpp
Parent: Child process completed.
```

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

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

    if (pid == 0) {
        execlp("ls", "ls", "-l", NULL);
        perror("execlp failed");
        exit(1);
    } else {
        wait(NULL);
        std::cout << "Parent: Child process completed." << std::endl;
    }

    return 0;
}
```

Q6.

```
~/Desktop/lab$ g++ q6.cpp -o q6
~/Desktop/lab$ ./q6
```

```
Sleeping now... (1s)
Sleeping now... (2s)
Sleeping now... (3s)
Sleeping now... (4s)
Sleeping now... (5s)
Alarm received. executing program...
```

```cpp
#include <iostream>
#include <unistd.h>
#include <signal.h>

void alarmHandler(int sig) {
    std::cout << "Alarm received. executing program..." << std::endl;
    exit(0);
}

int main() {
    signal(SIGALRM, alarmHandler);
    alarm(5);

    for (int i = 0; ; i++) {
        std::cout << "Sleeping now... (" << i + 1 << "s)" << std::endl;
        sleep(1);
    }

    return 0;
}
```

Q7.

```
~/Desktop/lab$ g++ q7.cpp -o q7
~/Desktop/lab$ ./q7
```

```
input.txt   q1    q1.cpp  q2.cpp  q3.cpp  q4.cpp  q5.cpp  q6.cpp  q7.cpp
output.txt  q1.c  q2      q3      q4      q5      q6      q7
Parent: Child process completed.
```

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

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

    if (pid == 0) {
        execlp("ls", "ls", NULL);
        perror("execlp failed");
        exit(1);
    } else {
        wait(NULL);
        std::cout << "Parent: Child process completed." << std::endl;
    }

    return 0;
}
```