Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4
за 4 семестр
По дисциплине: «ОСиСП»
Тема: «GCC. ПРОЦЕССЫ»

Выполнил:
студент 2-ого курса
Шевчук А. В.
группы ПО-3
Проверила:
Давидюк Ю. И.

Брест-2020

Задание для выполнения

Написать программу, которая будет реализовывать следующие функции:

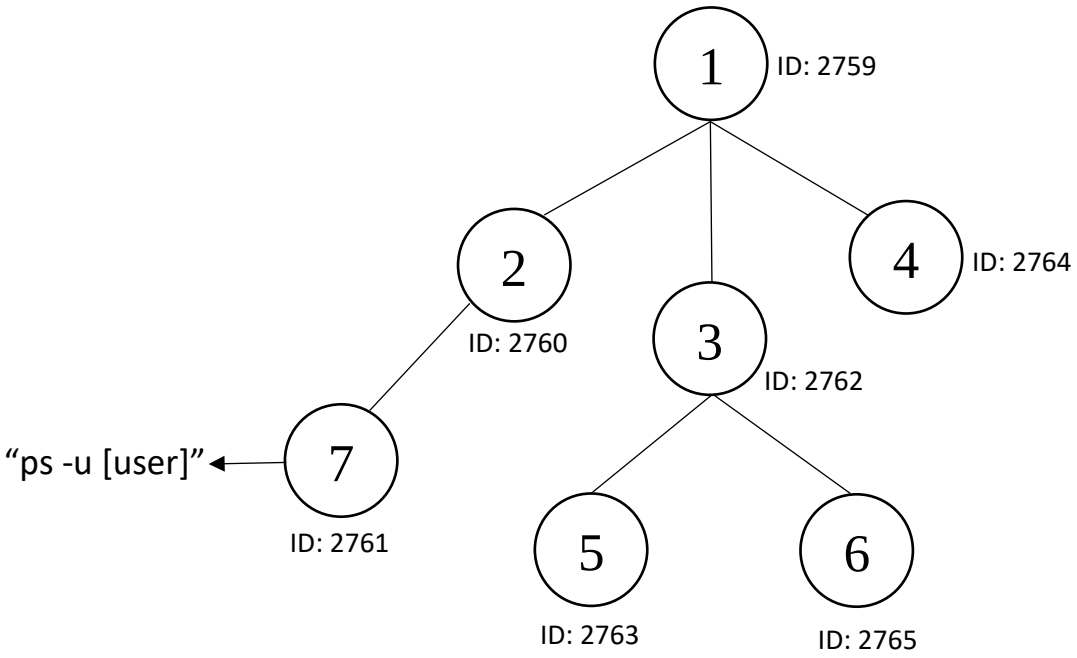
- сразу после запуска получает и сообщает свой ID и ID родительского процесса;
- перед каждым выводом сообщения об ID процесса и родительского процесса эта информация получается заново;
- порождает процессы, формируя генеалогическое дерево согласно варианту, сообщая, что "процесс с ID таким-то породил процесс с таким-то ID";
- перед завершением процесса сообщить, что "процесс с таким-то ID и таким-то ID родителя завершает работу";
- один из процессов должен вместо себя запустить программу, указанную в варианте задания.

На основании выходной информации программы предыдущего пункта изобразить генеалогическое дерево процессов (с указанием идентификаторов процессов). Объяснить каждое выведенное сообщение и их порядок в предыдущем пункте.

Ход работы

Вариант 27 (7)

№	fork	exec	
7	0 1 1 1 3 3 2	7	ps -u [user]



Текст программы:

lab4.c

```c
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
int main () {
        pid_t pid;
        printf("Порождение процесса 1: PID = %d, PPID = %d\n", getpid(), getppid());
        if ((pid = fork()) == -1)
                printf("Ошибка!\n");
        else if (pid == 0) {
                printf ("Порождение процесса 2: PID = %d, PPID = %d\n", getpid(), getppid());
                if ((pid = fork()) == -1)
                        printf("Ошибка!\n");
                else if (pid == 0) {
                        printf ("Порождение процесса 7: PID = %d, PPID = %d\n", getpid(), getppid());
                        printf ("Завершился процесс 7: PID = %d, PPID = %d\n", getpid(), getppid());
                        execl ("/bin/ps", "ps -u sergey", NULL);
                } else sleep(1);
                printf("Завершился процесс 2: PID = %d, PPID = %d\n", getpid(), getppid());
                exit(0);
        } else sleep(1);
        if ((pid = fork()) == -1)
                printf("Ошибка!\n");
        else if (pid == 0) {
                printf("Порождение процесса 3: PID = %d, PPID = %d\n", getpid(), getppid());
                if ((pid = fork()) == -1)
                        printf("Ошибка!\n");
                else if (pid == 0) {
                        printf("Порождение процесса 5: PID = %d, PPID = %d\n", getpid(), getppid());
                        printf("Завершился процесс 5: PID = %d, PPID = %d\n", getpid(), getppid());
                        exit(0);
                } else sleep(3);
                if ((pid = fork()) == -1)
                        printf("Ошибка!\n");
                else if (pid == 0) {
                        printf("Порождение процесса 6: PID = %d, PPID = %d\n", getpid(), getppid());
                        printf("Завершился процесс 6: PID = %d, PPID = %d\n", getpid(), getppid());
                        exit(0);
                } else sleep(4);
                printf("Завершился процесс 3: PID = %d, PPID = %d\n", getpid(), getppid());
                exit(0);
        } else sleep(2);
        if ((pid = fork()) == -1)
                printf("Ошибка!\n");
        else if (pid == 0) {
                printf("Порождение процесса 4: PID = %d, PPID = %d\n", getpid(), getppid());
                printf("Завершился процесс 4: PID = %d, PPID = %d\n", getpid(), getppid());
                exit (0);
        } else sleep(10);
        printf("Завершился процесс 1: PID = %d, PPID = %d\n", getpid(), getppid());
        exit(0);
        return 1;
}
```

```
sergey@sergey-VivoBook-15-ASUS-Laptop-X570UD:~/lab4$ gcc lab4.c -o lab4
sergey@sergey-VivoBook-15-ASUS-Laptop-X570UD:~/lab4$ ./lab4
Порождение процесса 1: PID = 2759, PPID = 2335
Порождение процесса 2: PID = 2760, PPID = 2759
Порождение процесса 7: PID = 2761, PPID = 2760
Завершился процесс 7: PID = 2761, PPID = 2760
  PID TTY          TIME CMD
 2335 pts/0    00:00:00 bash
 2759 pts/0    00:00:00 lab4
 2760 pts/0    00:00:00 lab4
 2761 pts/0    00:00:00 ps
Завершился процесс 2: PID = 2760, PPID = 2759
Порождение процесса 3: PID = 2762, PPID = 2759
Порождение процесса 5: PID = 2763, PPID = 2762
Завершился процесс 5: PID = 2763, PPID = 2762
Порождение процесса 4: PID = 2764, PPID = 2759
Завершился процесс 4: PID = 2764, PPID = 2759
Порождение процесса 6: PID = 2765, PPID = 2762
Завершился процесс 6: PID = 2765, PPID = 2762
Завершился процесс 3: PID = 2762, PPID = 2759
Завершился процесс 1: PID = 2759, PPID = 2335
```

В более раннем варианте программы задержка для процесса 2 отсутствовала:

```
        if ((pid = fork()) == -1)
                printf("Ошибка!\n");
        else if (pid == 0) {
                printf ("Порождение процесса 2: PID = %d, PPID = %d\n", getpid(), getppid());
                if ((pid = fork()) == -1)
                        printf("Ошибка!\n");
                else if (pid == 0) {
                        printf ("Порождение процесса 7: PID = %d, PPID = %d\n", getpid(), getppid());
                        printf ("Завершился процесс 7: PID = %d, PPID = %d\n", getpid(), getppid());
                        execl ("/bin/ps", "ps -u sergey", NULL);
                }
                printf("Завершился процесс 2: PID = %d, PPID = %d\n", getpid(), getppid());
                exit(0);
        } else sleep(10);
```

Это приводило к тому, что процесс 2 завершался раньше порождения процесса 7:

```
sergey@sergey-VivoBook-15-ASUS-Laptop-X570UD:~/lab4$ ./lab4
Порождение процесса 1: PID = 2488, PPID = 2316
Порождение процесса 2: PID = 2489, PPID = 2488
Завершился процесс 2: PID = 2489, PPID = 2488
Порождение процесса 7: PID = 2490, PPID = 2489
Завершился процесс 7: PID = 2490, PPID = 2489
  PID TTY          TIME CMD
 2316 pts/1    00:00:00 bash
```

Вывод: написал программу, работающую с процессами, которая сообщает ID процесса и ID родительского процесса. С помощью ID родителей и ID потомков удостоверился, что родители и их потомки соответствуют построенному генеалогическому древу процессов.