Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4
По дисциплине: «ОСиСП»
Тема: ««GCC. Процессы.»

Выполнил:
Студент 2 курса
Группы ПО-7
Комиссаров
А.Е.
Проверила:
Давидюк Ю.И.

2022

Цель: изучить работу с процессами на основе системы Linux.

Задание:

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 "процесса с ID таким-то породил процесса с таким-то ID";

• перед завершением процесса сообщить, что "процесса 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.

№	fork	exec	
6	0 1 1 2 4 4 4	6	ls

Код программы:

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
//execl("/usr/bin/ls", "ls", NULL);
//process 0
int main() {
    pid_t pid;
    printf("Process 0: PID = %d, PPID = %d\n", getpid(), getppid());
    //creating process 1
    if ((pid = fork()) == -1){
        printf("Error\n");
    }
    else if (pid == 0){
        printf("Process 1 created: PID = %d, PPID = %d\n", getpid(), getppid());
            //creating process 3
            if ((pid = fork()) == -1){
                printf("Error\n");
            }
            else if (pid == 0){
                printf("Process 3 created: PID = %d, PPID = %d\n", getpid(), getppid());
                    //creating process 4
                    if ((pid = fork()) == -1){
                        printf("Error\n");
                    }
                    else if (pid == 0){
                        printf("Process 4 created: PID = %d, PPID = %d\n", getpid(), getppid());
                        printf("Killing process 4: PID = %d, PPID = %d\n", getpid(), getppid());
```

```
                    exit(0);
                } else sleep(2);//killed process 4
                //creating process 5
                if ((pid = fork()) == -1){
                    printf("Error\n");
                }
                else if (pid == 0){
                    printf("Process 5 created: PID = %d, PPID = %d\n", getpid(), getppid());
                    printf("Killing process 5: PID = %d, PPID = %d\n", getpid(), getppid());
                    exit(0);
                } else sleep(2);//killed process 5
                //creating process 6
                if ((pid = fork()) == -1){
                    printf("Error\n");
                }
                else if (pid == 0){
                    printf("Process 6 created: PID = %d, PPID = %d\n", getpid(), getppid());
                    printf("Killing process 6: PID = %d, PPID = %d\n", getpid(), getppid());
                    execl("/usr/bin/ls", "ls", NULL);
                } else sleep(2);//killed process 6
            printf("Killing process 3: PID = %d, PPID = %d\n", getpid(), getppid());
            exit(0);
        } else sleep(7);//killed process 3
    printf("Killing process 1: PID = %d, PPID = %d\n", getpid(), getppid());
    exit(0);
} else sleep(8);//killed process 1
//creating process 2
if ((pid = fork()) == -1){
    printf("Error\n");
}
else if (pid == 0){
    printf("Process 2 created: PID = %d, PPID = %d\n", getpid(), getppid());
    printf("Killing process 2: PID = %d, PPID = %d\n", getpid(), getppid());
    exit(0);
}else sleep(2);

printf("Killing process 0: PID = %d, PPID = %d\n", getpid(), getppid());
return 0;
}
```

Результат работы программы:

```
/home/andrey/CLionProjects/untitled1/cmake-build-debug/untitled1
Process 0: PID = 3750, PPID = 1832
Process 1 created: PID = 3751, PPID = 3750
Process 3 created: PID = 3752, PPID = 3751
Process 4 created: PID = 3753, PPID = 3752
Killing process 4: PID = 3753, PPID = 3752
Process 5 created: PID = 3754, PPID = 3752
Killing process 5: PID = 3754, PPID = 3752
Process 6 created: PID = 3755, PPID = 3752
Killing process 6: PID = 3755, PPID = 3752
build.ninja CMakeFiles       Testing
CMakeCache.txt  cmake_install.cmake  untitled1
Killing process 3: PID = 3752, PPID = 3751
Killing process 1: PID = 3751, PPID = 3750
Process 2 created: PID = 3756, PPID = 3750
Killing process 2: PID = 3756, PPID = 3750
Killing process 0: PID = 3750, PPID = 1832

Process finished with exit code 0
```

Рис. 1 – результат работы программы

Визуальное представление дерева процессов:

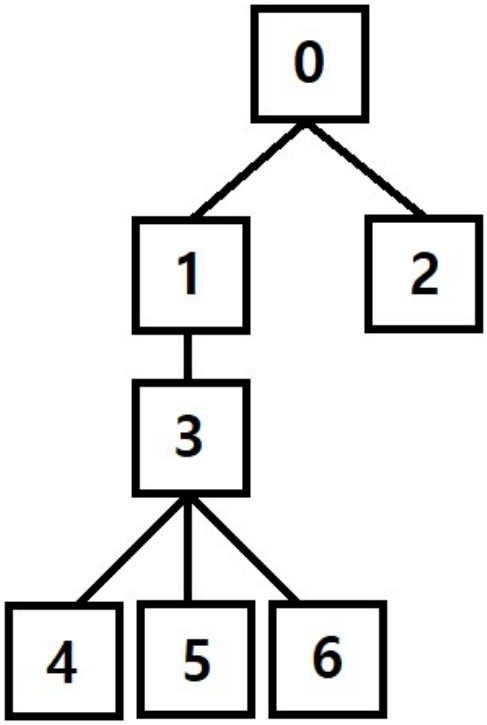

Рис. 2 – дерево процессов

Что происходит во время выполнения программы:

-Сначала порождается процесс 0.

-Потом порождается процесс 1 с родительским процессом 0.

-Порождается процесс 3 с родительским 1.

-Порождается 4 с родительским 3, завершается.

-Порождается 5 с родительским 3, завершается.

-Порождается 6 с родительским 3 (получаем информацию о содержимом каталога проекта CLion командой ls), завершается.

-Завершается процесс 3.

-Завершается процесс 1.

-Порождается процесс 2 с родительским 0, завершается.

-Завершается процесс 0.

Вывод: я изучил работу с процессами на примере системы Linux.