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>

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 2
        if ((pid = fork()) == -1) {
            printf("Error\n");
        } else if (pid == 0) {
            printf("Process 2 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());
                //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(1);//killed process 5
                    //creating process 5
                    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(1);//killed process 6
                    //creating process 6
                    if ((pid = fork()) == -1) {
                        printf("Error\n");
                    } else if (pid == 0) {
                        printf("Process 7 created: PID = %d, PPID = %d\n", getpid(),
        getppid());
                        printf("Killing process 7: PID = %d, PPID = %d\n", getpid(),
        getppid());
                        exit(0);
                    } else sleep(1);//killed process 7
                    printf("Killing process 4: PID = %d, PPID = %d\n", getpid(), getppid());
                    exit(0);
                } else sleep(4);//killed process 4
                printf("Killing process 2: PID = %d, PPID = %d\n", getpid(), getppid());
                exit(0);
            } else sleep(6);//killed process 2
            //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());
                printf("Killing process 3: PID = %d, PPID = %d\n", getpid(), getppid());
                exit(0);
            } else sleep(6);
            printf("Killing process 1: PID = %d, PPID = %d\n", getpid(), getppid());
            exit(0);
        }else sleep(13);
        printf("Killing process 0: PID = %d, PPID = %d\n", getpid(), getppid());
        return 0;
}
```

Результат работы программы:

```
Process 0: PID = 12190, PPID = 1466
Process 1 created: PID = 12191, PPID = 12190
Process 2 created: PID = 12192, PPID = 12191
Process 4 created: PID = 12193, PPID = 12192
Process 5 created: PID = 12194, PPID = 12193
Killing process 5: PID = 12194, PPID = 12193
Process 6 created: PID = 12195, PPID = 12193
Killing process 6: PID = 12195, PPID = 12193
build.ninja CMakeFiles       myfile.txt  untitled2
CMakeCache.txt  cmake_install.cmake  Testing
Process 7 created: PID = 12196, PPID = 12193
Killing process 7: PID = 12196, PPID = 12193
Killing process 4: PID = 12193, PPID = 12192
Killing process 2: PID = 12192, PPID = 12191
Process 3 created: PID = 12197, PPID = 12191
Killing process 3: PID = 12197, PPID = 12191
Killing process 1: PID = 12191, PPID = 12190
Killing process 0: PID = 12190, PPID = 1466

Process finished with exit code 0
```

Рис. 1 – результат работы программы

Визуальное представление дерева процессов:

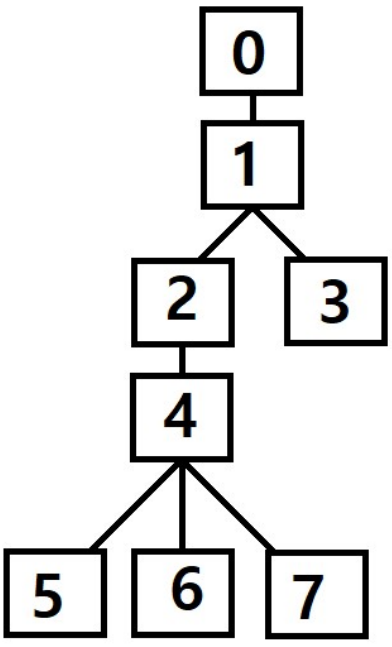

Рис. 2 – дерево процессов

Что происходит во время выполнения программы:

-Сначала порождается процесс 0.

-По очереди порождаются процессы 1(от 0), 2(от 1), 4(от 2), 5(от 4).

-Завершение 5-го процесса.

-Порождение 6(от 4) процесса с выполнением команды LS, завершение 6-го процесса.

-Порождение 7-го(от 4) и его завершение.

-Завершение 4-го процесса, после него 2-го процесса.

-Порождение 3-го(от 1) процесса и его завершение.

-Завершение 1-го, завершение 0-го.

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