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

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

Выполнил:
Студент 2 курса
Группы ПО-7(1)
Качан В.В.
Проверила:
Давидюк Ю. И.

Брест, 2022

Вариант 5

Цель работы: изучить работу с процессами в ОС 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	
5	0 1 1 2 2 3 3	5	df

Код программы:

```
#include <unistd.h>

#include <sys/types.h>

#include <stdio.h>

#include <stdlib.h>

int main() {

    pid_t pid;

    printf("Порождение процесса 0: PID = %d, PPID = %d\n", getpid(), getppid());

    //Порождение первого процесса

    if((pid = fork()) == -1){

        printf("Ошибка\n");

    } else if(pid == 0) {

        printf("Порождение процесса 1: PID = %d, PPID = %d\n", getpid(), getppid());

       //Порождение второго процесса

        if((pid = fork()) == -1){

            printf("Ошибка\n");

        } else if(pid == 0) {

            printf("Порождение процесса 2: PID = %d, PPID = %d\n", getpid(), getppid());
```

```
    //Порождение четвертого процесса
    if((pid = fork()) == -1){
        printf("Ошибка\n");
    } else if(pid == 0) {
        printf("Порождение процесса 4: PID = %d, PPID = %d\n", getpid(), getppid());
        printf("Завершение процесса 4: PID = %d, PPID = %d\n", getpid(), getppid());
        exit(0);
    } else sleep(1);
    //Порождение пятого процесса
    if((pid = fork()) == -1){
        printf("Ошибка\n");
    } else if(pid == 0) {
        printf("Порождение процесса 5: PID = %d, PPID = %d\n", getpid(), getppid());
        printf("Завершение процесса 5: PID = %d, PPID = %d\n", getpid(), getppid());
        execl("/usr/bin/df","df", NULL);
    } else sleep(1);
    printf("Завершение процесса 2: PID = %d, PPID = %d\n", getpid(), getppid());
    exit(0);
} else sleep(4);
//Порождение третьего процесса
if((pid = fork()) == -1){
    printf("Ошибка\n");
} else if(pid == 0) {
    printf("Порождение процесса 3: PID = %d, PPID = %d\n", getpid(), getppid());
    //Порождение шестого процесса
    if((pid = fork()) == -1){
        printf("Ошибка\n");
    } else if(pid == 0) {
        printf("Порождение процесса 6: PID = %d, PPID = %d\n", getpid(), getppid());
        printf("Завершение процесса 6: PID = %d, PPID = %d\n", getpid(), getppid());
        exit(0);
    } else sleep(1);
    //Порождение седьмого процесса
    if((pid = fork()) == -1){
        printf("Ошибка\n");
```

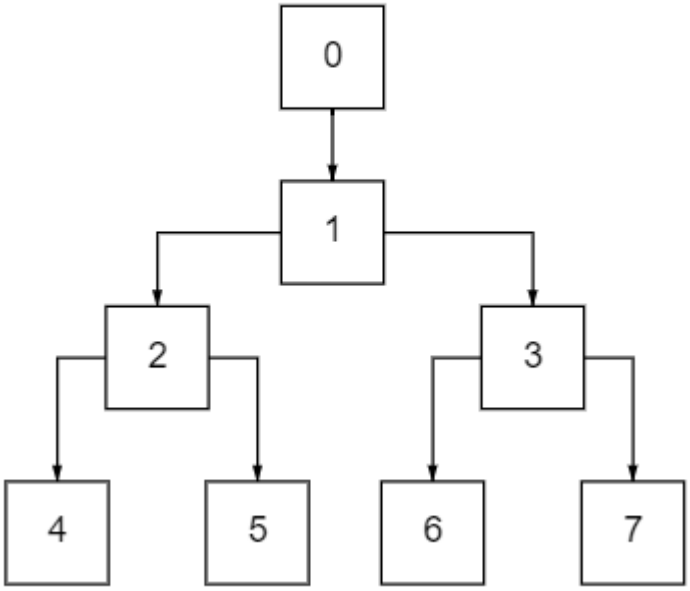
```
            } else if(pid == 0) {
                printf("Порождение процесса 7: PID = %d, PPID = %d\n", getpid(), getppid());
                printf("Завершение процесса 7: PID = %d, PPID = %d\n", getpid(), getppid());
                exit(0);
            } else sleep(1);
            printf("Завершение процесса 3: PID = %d, PPID = %d\n", getpid(), getppid());
            exit(0);
        } else sleep(4);
        printf("Завершение процесса 1: PID = %d, PPID = %d\n", getpid(), getppid());
        exit(0);
    } else sleep(10);
    printf("Завершение процесса 0: PID = %d, PPID = %d\n", getpid(), getppid());

    return 0;
}
```

Результат выполнения:

```
Порождение процесса 0: PID = 5412, PPID = 4273
Порождение процесса 1: PID = 5413, PPID = 5412
Порождение процесса 2: PID = 5414, PPID = 5413
Порождение процесса 4: PID = 5415, PPID = 5414
Завершение процесса 4: PID = 5415, PPID = 5414
Порождение процесса 5: PID = 5416, PPID = 5414
Завершение процесса 5: PID = 5416, PPID = 5414
Filesystem     1K-blocks     Used Available Use% Mounted on
udev             3159220        0   3159220   0% /dev
tmpfs             638540     1356    637184   1% /run
/dev/sda5       19992176 11661176   7292408  62% /
tmpfs            3192688    42944   3149744   2% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            3192688        0   3192688   0% /sys/fs/cgroup
/dev/loop0           128      128         0 100% /snap/bare/5
/dev/loop2         63488    63488         0 100% /snap/core20/1405
/dev/loop1        821248   821248         0 100% /snap/clion/189
/dev/loop3        254848   254848         0 100% /snap/gnome-3-38-2004/99
/dev/loop4         63488    63488         0 100% /snap/core20/1376
/dev/loop5         55552    55552         0 100% /snap/snap-store/558
/dev/loop7         45824    45824         0 100% /snap/snapd/15314
/dev/loop6         44800    44800         0 100% /snap/snapd/15177
/dev/loop8         66816    66816         0 100% /snap/gtk-common-themes/1519
/dev/loop9         56960    56960         0 100% /snap/core18/2344
/dev/sda1         523248        4    523244   1% /boot/efi
tmpfs             638536       44    638492   1% /run/user/1000
Завершение процесса 2: PID = 5414, PPID = 5413
Порождение процесса 3: PID = 5417, PPID = 5413
Порождение процесса 6: PID = 5418, PPID = 5417
Завершение процесса 6: PID = 5418, PPID = 5417
Порождение процесса 7: PID = 5419, PPID = 5417
Завершение процесса 7: PID = 5419, PPID = 5417
Завершение процесса 3: PID = 5417, PPID = 5413
Завершение процесса 1: PID = 5413, PPID = 5412
Завершение процесса 0: PID = 5412, PPID = 4273
```

Генеалогическое дерево процессов:



Сначала порождается процесс 0. Потом порождается процесс 1 с родительским процессом 0. Порождается процесс 2 с родительским 1. Порождается 4 с родительским 2, завершается. Порождается 5 с родительским 2, завершается. Получаем полную сводку об использовании доступного и используемого дискового пространства файловой системы. Завершается процесс 2. Порождается 3 с родительским 2. Порождается 6 с родительским 3, завершается. Порождается 7 с родительским 3, завершается. Завершается процесс 3. Завершается процесс 1. Завершается процесс 0.

Вывод: изучил работу с процессами в ОС Linux.