Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

по дисциплине «Операционные системы и системное программирование»

Тема: «GCC. Процессы»

Вариант 7

Выполнил:

студент 2-го курса

группы ПО-6

Лавренчик Д.О.

Проверил:

Давидюк Ю.И.

Брест 2022

**Лабораторная работа №4**

**Тема:** «GCC. Процессы»

**Цель работы:** ознакомиться с компиляторо GCC; изучить создание, завершение и изменение пользовательского контекста процессов в операционной системе UNIX.

**Ход работы**

***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.

***Вариант индивидуального задания:***

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *fork* | *exec* | |
| 7 | 0111332 | 7 | ps –u [user] |

**Текст программы:**

#include <unistd.h>

#include <sys/types.h>

#include <stdio.h>

#include <stdlib.h>

int main()

{

pid\_t pid;

printf("Процесс 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 7 PID = %d PPID = %d\n",getpid(), getppid());

printf("Завершился процесс 7 PID = %d PPID = %d\n",getpid(), getppid());

execl("/bin/ps","ps -u [derelya]",NULL);

exit(0);

}

else sleep(1);

printf("Завершился процесс 2 PID = %d PPID = %d\n", getpid(), getppid());

exit(0);

}

else sleep(3);//задержка родительского процесса (2 процесс sleep)

//Порождение 3 процесса

if ((pid = fork()) == -1) { printf("Ошибка!\n"); }

else if (pid == 0)

{

printf("Порождение процесса 3 PID = %d PPID = %d\n", getpid(), getppid());

// 5

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(1);

//Порождение 6 процесса

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(1);

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(1);

printf("Завершился процесс 1 PID = %d PPID = %d\n", getpid(), getppid());

exit(0);

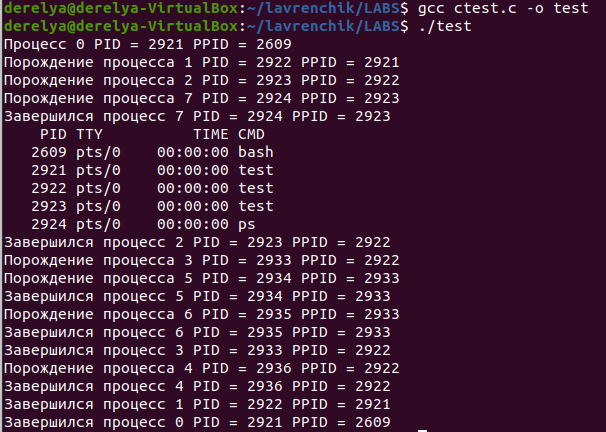
}

else sleep(7);

printf("Завершился процесс 0 PID = %d PPID = %d\n", getpid(), getppid());

return 1;

}**Результат выполнения программы:**



**Генеалогическое дерево процессов:**

**0**

**1**

**2**

**3**

**4**

**5**

**7**

**6**

Процесс 0 не имеет родителя. В начале программы процессом 0 порождается процесс 1. Затем порождается процесс 2, родителем которого является процесс 1. Далее порождается и завершается процесс 7, родитель которого — процесс 2. Процесс 7 вместо себя запускает программу ps -u [derelya]. Процессом 1 порождается процесс 3, который становится родителем для процесса 5 и 6, которые сразу же завершаются. Далее процессом 1 порождается процесс 4 и завершается, а за ним — процесс 1. Последним завершается процесс 0.

**Вывод:** в ходе выполнения данной лабораторной работы были изучены основы создания, завершения и изменения пользовательского контекста процессов в операционной системе UNIX.