Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

за 4 семестр

По дисциплине: «ОСиСП»

Тема: «GCC.Процессы»

Выполнила:

Студентка 2 курса

Группы ПО-3(1

Давидюк Ю. И.

Выполнила:

Студентка 2 курса

Группы ПО-3(1)

Гаврилюк Р.И.

Проверил:

Давидюк Ю. И

Выполнил:

Студент 2 курса

Группы ПО-4(2)

Кречко К.А.

Проверила:

Давидюк Ю.И.

Брест 2021

GCC. Процессы

**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.

**Вариант 14**



**Код:**

#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());

//Порождение процесса 2

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

printf("Ошибка!\n");

else if (pid == 0) {

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

//Порождение процесса 3

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

printf("Ошибка!\n");

else if (pid == 0) {

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

//Порождение процесса 5

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());

exit(0);

} else sleep(1);

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

exit(0);

} else sleep(2);

//Порождение процесса 4

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

printf("Ошибка!\n");

else if (pid == 0) {

printf("Порождение процесса 4 PID = %d PPID = %d\n", getpid(), getppid());

//Порождение процесса 6

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

printf("Ошибка!\n");

else if (pid == 0) {

printf("Порождение процесса 6 PID = %d PPID = %d\n", getpid(), getppid());

//Порождение процесса 7

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());

execl("/bin/who","who",NULL);

exit(0);

} else sleep(3);

printf("Завершение процесса 6 PID = %d PPID = %d\n", getpid(), getppid());

exit(0);

} else sleep(4);

printf("Завершение процесса 4 PID = %d PPID = %d\n", getpid(), getppid());

exit(0);

} else sleep(7);

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

exit(0);

}else sleep(9);

sleep(1);

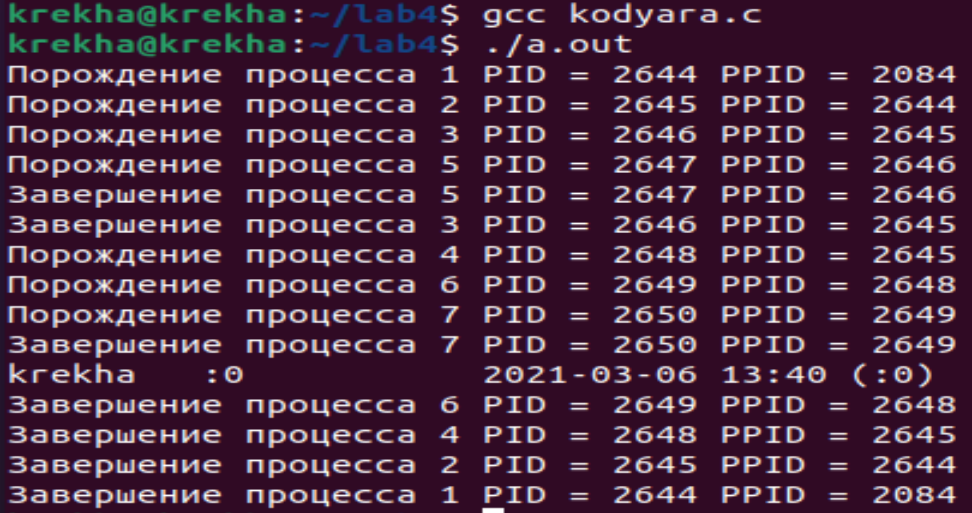
printf("Завершение процесса 1 PID = %d PPID = %d\n", getpid(), getppid());

exit(0);

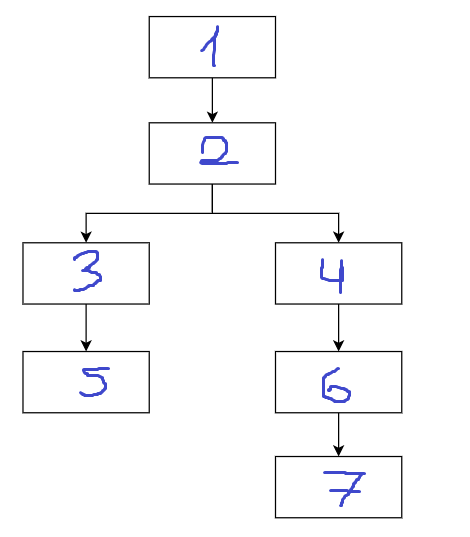
return 1;

}

**Результат:**

****

**Генеалогическое дерево процессов:**



Сначала порождается процесс 1. От него порождается процесс 2. От родительского процесса 2 порождается процесс 3. От родительского процесса 3 порождаем процесс 5. Завершаем процесс 5. Затем завершаем процесс 3. От родительского процесса 2 порождаем процесс 4. От родительского процесса 4 порождаем процесс 6. От родительского процесса 6 порождаем процесс 7. Выполняем who. Завершаем процессы в следующем порядке: 7,6,4,2,1.

**Вывод:** Я изучил работу с процессами в ОС Linux.