Министерство образования Республики Беларусь

Учреждение образования

“Брестский Государственный технический университет”

Кафедра ИИТ

Лабораторная работа №4

По дисциплине “Операционные системы”

Тема: “GCC. Процессы”

Выполнил:

Студент 2 курса

Группы ИИ-23

Вышинский А. С.

Проверил:

Чеслов А. В.

Брест 2023

**Цель работы:** ознакомится с работой компилятора gcc. А также изучить функции для работы с процессами в линуксе.

**Ход работы:**

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 3

#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 4 PID = %d PPID = %d\n",getpid(), getppid());

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

exit(0);

} else sleep(0.5);

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

exit(0);

}else sleep(1);

// Порождение третьего процесса

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

printf("Ошибка!\n");

else if (pid == 0) {

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

//порождение пятого цикла

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

printf("Ошибка!\n");

else if (pid == 0) {

printf("Порождение 5 PID = %d PPID = %d\n", getpid(), getppid());

printf("Завершился процесс 5: PID = %d, PPID = %d\n",getpid(), getppid());

exit(0);

} else sleep(0.25);

//порождение шестого цикла

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

printf("Ошибка!\n");

else if (pid == 0) {

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

//порождение седьмого цикла

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

printf("Ошибка!\n");

else if (pid == 0) {

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

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

exit(0);

}

printf("Завершился процесс 6: PID = %d, PPID = %d\n", getpid(), getppid());

exit(0);

} else sleep(0.25);

printf("Завершился процесс 3: PID = %d, PPID = %d\n",

getpid(), getppid());

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

}else sleep(1.5);

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

getppid());

exit(0);

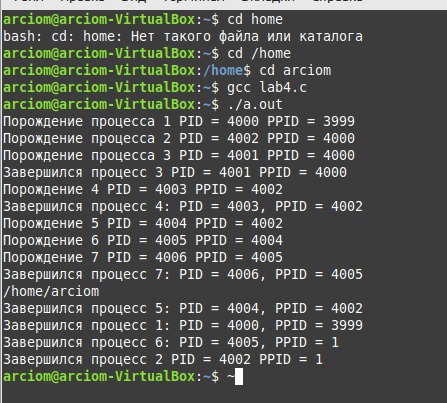
} else sleep(4);

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

getppid());

return 0;

}

**Вывод:** ознакомился с работой компилятора gcc. А также изучил функции для работы с процессами в линуксе.