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(2)

Овсейчук В.И.

Проверил:

Давидюк Ю.И.

2020

Лабораторная работа №4

Вариант 19

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).



#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 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(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());

// 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("Завершился процесс 6 PID = %d PPID = %d\n",

getpid(), getppid());

exit(0);

}

else sleep(4);

// 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("Завершился процесс 7 PID = %d PPID = %d\n",

getpid(), getppid());

exit(0);

}

else sleep(3);

printf("Завершился процесс 5 PID = %d PPID = %d\n",

getpid(), getppid());

execl("/usr/bin/whoami","whoami", NULL);

}

else sleep(8);

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

getppid());

exit(0);

}

else sleep(4); //задержка родительского процесса

// 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());

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

getppid());

exit(0);

}

else sleep(7);

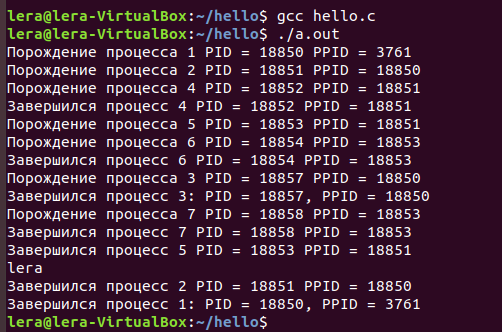
printf("Завершился процесс 1: PID = %d, PPID = %d\n", getpid(),

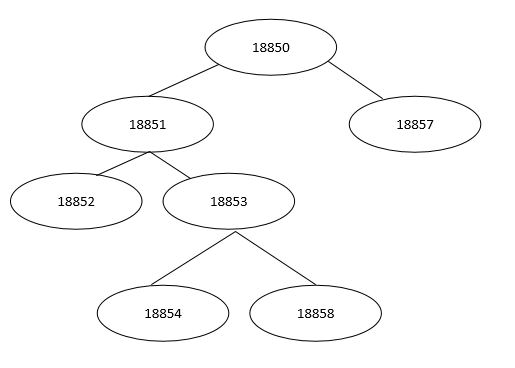
getppid());

exit(0);

return 1;

}





**Вывод:** изучила GCC. процессы.