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-21

Литвинюк Т. В.

**Проверил:**

Козинский А. А.

Брест 2022

**Цель:** знакомство с компилятором GCC, изучение процессов в Linux.

**Ход работы:**

**Вариант 8**

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. 

#include <unistd.h>

#include <sys/types.h>

#include <stdio.h>

#include <stdlib.h>

int main(){

pid\_t pid;

printf("PID: %d, PPID: %d\n", getpid(), getppid());

if ((pid = fork()) < 0){ // 2

printf("ERROR:2\n");

exit(1);

}

else if (pid == 0){

printf("Proccess 2:%d was born by proccess %d.\n", getpid(), getppid());

if ((pid = fork()) < 0){ // 3

printf("ERROR:3\n");

exit(1);

}

else if (pid == 0){

printf("Proccess 3:%d was born by proccess %d.\n", getpid(), getppid());

if ((pid = fork()) < 0){ // 5

printf("ERROR:5\n");

exit(1);

}

else if (pid == 0){

printf("Proccess 5:%d was born by proccess %d.\n", getpid(), getppid());

printf("Proccess 5:%d was closed. PPID : %d.\n", getpid(), getppid());

exit(0);

} else sleep(1);

if ((pid = fork()) < 0){ // 7

printf("ERROR:7\n");

exit(1);

}

else if (pid == 0){

printf("Proccess 7:%d was born by proccess %d.\n", getpid(), getppid());

printf("Proccess 7:%d was closed. PPID : %d.\n", getpid(), getppid());

exit(0);

} else sleep(1);

printf("Proccess 3:%d was closed. PPID : %d.\n", getpid(), getppid());

exit(0);

} else sleep(1);

if ((pid = fork()) < 0){ // 4

printf("ERROR:4\n");

exit(1);

}

else if (pid == 0){

printf("Proccess 4:%d was born by proccess %d.\n", getpid(), getppid());

if ((pid = fork()) < 0){ // 6

printf("ERROR:6\n");

exit(1);

}

else if (pid == 0){

printf("Proccess 6:%d was born by proccess %d.\n", getpid(), getppid());

printf("Proccess 6:%d was closed. PPID : %d.\n", getpid(), getppid());

exit(0);

} else sleep(1);

printf("Proccess 4:%d was closed. PPID : %d.\n", getpid(), getppid());

exit(0);

} else sleep(1);

printf("Proccess 2:%d was closed. PPID : %d.\n", getpid(), getppid());

exit(0);

}

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

}//0122345

[tim@fedora lab4]$ ./a.out

PID: 2138, PPID: 2097

Proccess 2:2139 was born by proccess 2138.

Proccess 3:2140 was born by proccess 2139.

Proccess 5:2141 was born by proccess 2140.

Proccess 5:2141 was closed. PPID : 2140.

/mnt/Files/Studing/OS/lab4

Proccess 4:2146 was born by proccess 2139.

Proccess 7:2147 was born by proccess 2140.

Proccess 7:2147 was closed. PPID : 2140.

Proccess 6:2148 was born by proccess 2146.

Proccess 6:2148 was closed. PPID : 2146.

Proccess 2:2139 was closed. PPID : 1301.

Proccess 3:2140 was closed. PPID : 2139.

Proccess 4:2146 was closed. PPID : 2139.

**Вывод:** в ходе лабораторной работы я научился работать с компилятором gcc и с процессами в Linux.