Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

за 4 семестр

По дисциплине: «ОСиСП»

Тема: «GCC. Процессы»

Выполнил:

Студент 2 курса

Группы ПО-4(1)

Иваненко И. Л.

Проверил:

Давидюк Ю. И.

2021

Лабораторная работа №4

GCC. Процессы

Цель работы: изучить работу с процессами в ОС Linux.

Задание для выполнения

Вариант 10

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.

Код программы:

#include <unistd.h>

#include <sys/types.h>

#include <stdio.h>

#include <stdlib.h>

int main() {

pid\_t pid;

// Process 1 spawn

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

// Process 2 spawn

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

printf("Error!\n");

} else if(pid == 0) {

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

sleep(3);

// Process 5 spawn

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

printf("Error!\n");

} else if(pid == 0) {

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

// Process 6 spawn

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

printf("Error!\n");

} else if(pid == 0) {

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

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

exit(0);

} else sleep(1);

// Process 6 dies

// Process 7 spawn

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

printf("Error!\n");

} else if(pid == 0) {

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

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

exit(0);

} else sleep(1);

// Process 7 dies

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

exit(0);

} else sleep(3);

// Process 5 dies

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

exit(0);

// Process 5 dies

} else sleep(1);

// Process 2 dies

// Process 3 spawn

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

printf("Error!\n");

} else if(pid == 0) {

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

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

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

} else sleep(1);

// Process 3 dies

// Process 4 spawn

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

printf("Error!\n");

} else if(pid == 0) {

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

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

exit(0);

} else sleep(1);

// Process 4 dies

sleep(4);

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

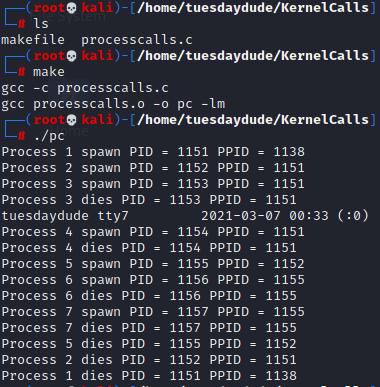
exit(0);

// Process 1 dies

return 1;

}

Результат выполнения:



Генеалогическое дерево процессов:



Сначал порождается процесс 1, затем порождается процесс 2 с родительским процессом 1, порождается процесс 3 с родительским процессом 1, завершается процесс 3, порождается процесс 4 с родительским процессом 1, завершается процесс 4. Порождается процесс 5 с родительским процессом 2. Порждается процесс 6 с родительским процессом 5, завершается процесс 6, порождается процесс 7 с родителським процессом 5, завершается процесс 7, завершается процесс 5, завершается процесс 2, завершается процесс 1.

Вывод: в ходе выполнения данной работы изучил работу с процессами в ОС Linux.