**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №6**

**по дисциплине «Организация процессов и программирование в среде Linux»**

Темы: ОРГАНИЗАЦИЯ ПЕРИОДИЧЕСКИХ ПРОЦЕССОВ

|  |  |  |
| --- | --- | --- |
| Студент гр. 8306 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Пеунов В.В. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Разумовский Г.В. |

Санкт-Петербург,

2021

**Цель работы**

Использование сервиса cron, механизма сигналов и интервальных таймеров для организации периодических процессов.

**Задание на лабораторную работу**

1. Написать периодическую программу, в которой период запуска и количество запусков должны задаваться в качестве ее параметров. При каждом очередном запуске программа должна порождать новый процесс, который выводить на экран свой идентификатор, дату и время старта. Программа и ее дочерний процесс должны быть заблокированы от завершения при нажатии клавиши Ctrl/z. После завершения дочернего процесса программа должна вывести на экран информацию о времени своей работы и дочернего процесса

2. Откомпилировать программу и запустить ее несколько раз с разным периодом запуска и количеством повторений

**Описание работы**

После написания программы было проведено 3 эксперимента. С разными значениями задержек.

|  |  |  |
| --- | --- | --- |
|  | Количество запусков | Задержка |
| Эксперимент 1 | 3 | 2 |
| Эксперимент 2 | 5 | 1 |
| Эксперимент 3 | 2 | 5 |

**Эксперимент 1**

Потомок: №1

Process ID: 106902

Время старта: Mon Nov 1 19:56:19 2021

Время работы: 4162 тиков

Потомок: №2

Process ID: 106903

Время старта: Mon Nov 1 19:56:21 2021

Время работы: 4557 тиков

Потомок: №3

Process ID: 106911

Время старта: Mon Nov 1 19:56:23 2021

Время работы: 4989 тиков

Родитель

Process ID: 106900

Время старта: Mon Nov 1 19:56:17 2021

Время работы: 6 секунд

**Эксперимент 2**

Потомок: №1

Process ID: 106954

Время старта: Mon Nov 1 19:56:51 2021

Время работы: 3981 тиков

Потомок: №2

Process ID: 106955

Время старта: Mon Nov 1 19:56:52 2021

Время работы: 4379 тиков

Потомок: №3

Process ID: 106958

Время старта: Mon Nov 1 19:56:53 2021

Время работы: 4812 тиков

Потомок: №4

Process ID: 106974

Время старта: Mon Nov 1 19:56:54 2021

Время работы: 5235 тиков

Потомок: №5

Process ID: 106975

Время старта: Mon Nov 1 19:56:55 2021

Время работы: 5695 тиков

Родитель

Process ID: 106953

Время старта: Mon Nov 1 19:56:50 2021

Время работы: 5 секунд

**Эксперимент 3**

Потомок: №1

Process ID: 107241

Время старта: Mon Nov 1 19:59:56 2021

Время работы: 4034 тиков

Потомок: №2

Process ID: 107244

Время старта: Mon Nov 1 20:00:01 2021

Время работы: 4406 тиков

Родитель

Process ID: 107236

Время старта: Mon Nov 1 19:59:51 2021

Время работы: 10 секунд

**Вывод:** в ходе лабораторной работы были отработаны навыки работы с механизмом сигналов и получены компетенции работы с интервальными таймерами для организации периодических процессов.

**Приложение A. Основная программа**

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <string>

using namespace std;

int mainStop = 10;

int childFirstStop = 5;

int childSecondStop = 3;

const char \*childFirstProgramPath = "/home/peunov/highschool/opp-linux/laba1.2/cmake-build-debug/laba1\_2";

void outputInFile(string processName, string filePath, int stop);

int error();

string getFilePath();

int main(int argc, char\*\* argv){

if(argc == 4){

mainStop = atoi(argv[1]);

childFirstStop = atoi(argv[2]);

childSecondStop = atoi(argv[3]);

}

string filePath = getFilePath();

pid\_t fork\_process\_id = fork();

if(fork\_process\_id == -1){

return error();

}

if(fork\_process\_id == 0){

sleep(childFirstStop);

outputInFile("Потомок 1", filePath, childFirstStop);

exit(EXIT\_SUCCESS);

}

if(fork\_process\_id > 0){

pid\_t vfork\_process\_id = vfork();

if(vfork\_process\_id == -1){

return error();

}

if(vfork\_process\_id > 0){

sleep(mainStop);

outputInFile("Родитель", filePath, mainStop);

exit(EXIT\_SUCCESS);

}

if(vfork\_process\_id == 0){

sleep(childSecondStop);

outputInFile("Потомок 2", filePath, childSecondStop);

execlp(childFirstProgramPath, nullptr);

exit(EXIT\_SUCCESS);

}

}

return 0;

}

void outputInFile(string processName, string filePath, int stop){

ofstream file;

file.open(filePath, ios::app);

if (file.is\_open()) {

pid\_t process\_id = getpid();

file << processName << " - задержка: " << stop << endl;

file << processName << " - идентификатор процесса: " << process\_id << endl;

file << processName << " - идентификатор предка: " << getppid() << endl;

file << processName << " - идентификатор сессии процесса: " << getsid(process\_id) << endl;

file << processName << " - идентификатор группы процессов: " << getpgid(process\_id) << endl;

file << processName << " - реальный идентификатор пользователя: " << getuid() << endl;

file << processName << " - эффективный идентификатор пользователя: " << geteuid() << endl;

file << processName << " - реальный групповой идентификатор: " << getgid() << endl;

file << processName << " - эффективный групповой идентификатор: " << getegid() << endl;

}

file.close();

}

int error(){

cout << "При создании процесса произошла ошибка";

return 1;

}

string getFilePath(){

string filePath;

cout << "Введите путь к файлу: ";

cin >> filePath;

return filePath;

}

**Приложение B. Заменяемая программа**

#include <string>

int main() {

exit(EXIT\_SUCCESS);

}