1. МИНОБРНАУКИ РОССИИ
2. САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
3. ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
4. «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
5. Кафедра Вычислительной техники

ОТЧЁТ

по лабораторной работе №6

по дисциплине «Организация процессов и программирования в среде Linux»

1. Тема: ОРГАНИЗАЦИЯ ПЕРИОДИЧЕСКИХ ПРОЦЕССОВ

|  |  |  |
| --- | --- | --- |
| Студент гр. 1335 |  | Максимов Ю Е |
| Преподаватель |  | Разумовский Г.В. |

Оглавление

[1. Введение 3](#__RefHeading___Toc336_311249038)

[1.1. Введение 3](#__RefHeading___Toc15338_3795591338)

[1.2. Порядок выполнения работы 3](#__RefHeading___Toc15340_3795591338)

[1.3. Содержание отчёта 4](#__RefHeading___Toc15342_3795591338)

[2. Текст периодической программы 5](#__RefHeading___Toc2116_1927017605)

[3. Скриншот экрана результатов работы периодической программы после всех её перезапусков 8](#__RefHeading___Toc2118_1927017605)

[4. Вывод 10](#__RefHeading___Toc358_311249038)

[5. Список использованных источников 11](#__RefHeading___Toc360_311249038)

# 1. Введение

## 1.1. Введение

Тема работы: Организация периодических процессов.

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

## 1.2. Порядок выполнения работы

1. Создать пользовательский файл конфигурации сервиса cron, в котором содержатся команды периодического запуска одной из программ, разработанных в предыдущих лабораторных работах. Результаты работы этой программы должны выводиться или переадресовываться в файл.

2. После нескольких запусков программы удалить пользовательский файл конфигурации.

3. Написать периодическую программу, в которой период запуска и количество запусков должны задаваться в качестве её параметров. При каждом очередном запуске программа должна порождать новый процесс, который выводит на экран свой идентификатор, дату и время старта. Программа и её 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.

4. Откомпилировать программу и запустить её несколько раз с разным периодом запуска и количеством повторений.

Выбранные задания: 3, 4.

## 1.3. Содержание отчёта

Отчёт по лабораторной работе должен содержать:

1. Цель и задание.

2. Распечатки файлов расписания и результатов работы программы.

3. Текст периодической программы.

4. Скриншот экрана результатов работы периодической программы после всех её перезапусков.

# 2. Текст периодической программы

// start program

// ./main <launch period> <number of launches>

// <launch period> -- launch period in seconds (time between every launch)

// <number of launches> -- number of program launches (how many times program will launch)

#include <iostream>

#include <signal.h>

#include <sys/wait.h>

#include <sys/time.h>

#include <unistd.h>

using namespace std;

void processFunction (int local\_int); // function for repeat

float finalTime (float local\_start\_time); // counting final time

int main(int argc, char \*argv[])

{

int number\_period = 0; // launch period, 0 by default

int number\_launch = 0; // number of launches, 0 by default

float start\_time; // parent process start time

struct itimerval value; // for timer structure, new time set

struct itimerval old\_value; // for timer structure, old timer dropped here

struct sigaction sigact;

number\_period = atoi(argv[1]); // convert input data to numeric

number\_launch = atoi(argv[2]); // convert input data to numeric

//start\_time = (float)clock() / CLOCKS\_PER\_SEC; // measuring the current time

sigact.sa\_handler = processFunction; // setting handling function

sigemptyset(&sigact.sa\_mask); // cleaning "set" from all signals

sigaddset(&sigact.sa\_mask, SIGTSTP); // SIGTSTP signal adding to mask

sigprocmask(SIG\_BLOCK, &sigact.sa\_mask, NULL); // SIGTSTP signal blocking

sigact.sa\_flags = 0; // no flags added

sigaction(SIGALRM, &sigact, NULL); // setting responce to SIGALRM signal

// settings for 1st launch, because time (10 us) will be left very fast

value.it\_value.tv\_sec = 0; // setting 0 seconds. only useconds

value.it\_value.tv\_usec = 10; // setting 10 useconds for fast timer counting --> faster the signal will be sended

// setting new interval

value.it\_interval.tv\_sec = number\_period; // interval s, launch period

value.it\_interval.tv\_usec = 0; // interval us, launch period

setitimer(ITIMER\_REAL, &value, &old\_value); // ITIMER\_REAL means always

for (int i = 0; i < number\_launch; i++)

{

start\_time = (float)clock() / CLOCKS\_PER\_SEC; // measuring the current time

pause(); // SIGALRM signal waiting

cout << "Parent process' work time (seconds): " << finalTime(start\_time) << "\n\n";

}

return 0;

}

void processFunction (int local\_int) // function for repeat

{

int local\_status; // status for "waitpid" func

float local\_start\_time; // initilaizing

pid\_t local\_pid = fork(); // creating child process

if (local\_pid == 0) // if child is created

{

time\_t local\_seconds = time (NULL); // seconds since

sigset\_t local\_set; // signal set

struct tm\* local\_time = localtime (&local\_seconds); // time date

local\_start\_time = (float)clock() / CLOCKS\_PER\_SEC; // counting current time since starting program

sigemptyset(&local\_set); // setting emty signal set

cout << "Child process' PID: " << getpid() << "\n"; // get child process' parent ID

cout << "Parent process' work start time: " << asctime(local\_time); //<< "\n";

sigaddset(&local\_set, SIGTSTP); // adding signal SIGTSTP (anti ctrl-z) to the current process

sigprocmask(SIG\_BLOCK, &local\_set, NULL); // adding blocked signals to the set, SIG\_BLOCK means blocked signals are current set + set argument in function

//cout << "Child process' work time (seconds): " << finalTime(local\_start\_time) << "\n"; // cout child process time

exit(EXIT\_SUCCESS);

}

else // else if it is not child or child is not created

{

waitpid(local\_pid, &local\_status, 0); // wait until child is terminate its work

cout << "Child process' work time (seconds): " << finalTime(local\_start\_time) << "\n"; // cout child process time

}

}

float finalTime (float local\_start\_time) // counting current time since starting prohram & last function

{

//float local\_end\_time = 0;

//local\_end\_time = ((float)clock()) / CLOCKS\_PER\_SEC);

return (((float)clock() / CLOCKS\_PER\_SEC) - local\_start\_time);

}

# 3. Скриншот экрана результатов работы периодической программы после всех её перезапусков

Программа запускается с помощью команды «./main <период запуска> <количество запусков>», где <период запуска> – это период, который проходит между запусками программы и который измеряется в секундах, и где <количество запусков> – это количество раз, которое запускается программа.

При запуске программы с количество запусков, равным 0, программа не запустится. При запуске программы с периодом запуска, равным 0, программа будет запускаться.

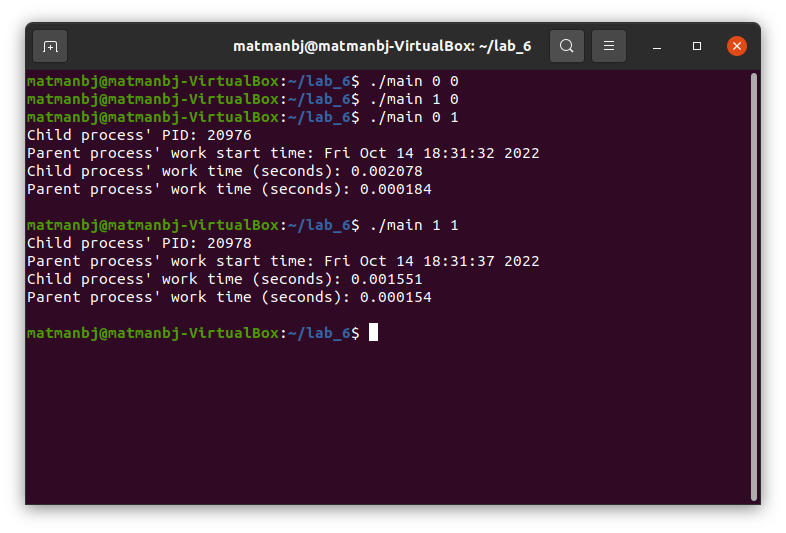


Рисунок 1. Запуск программы с периодом запуска и количеством раз, равными «0 0», «1 0», «0 1» и «1 1» соответственно

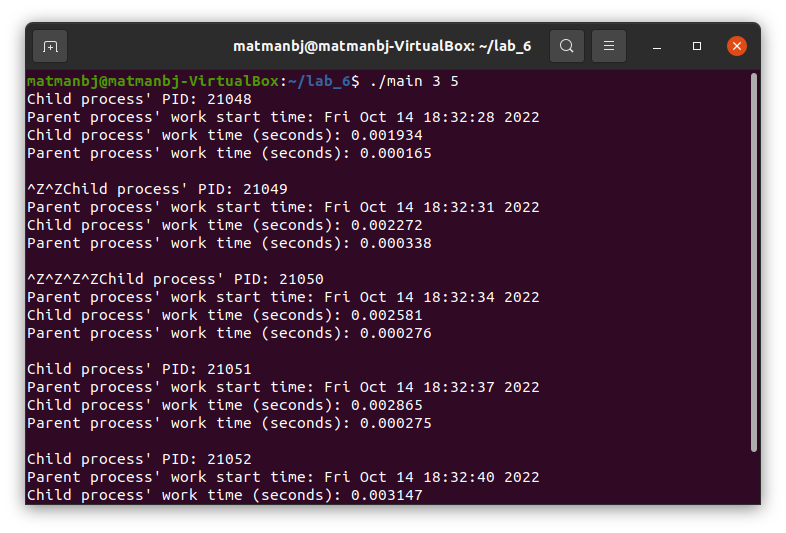


Рисунок 2. Запуск программы с периодом запуска и количеством раз, равными «3 5» соответственно, с нажатием клавиш «Ctrl-Z»

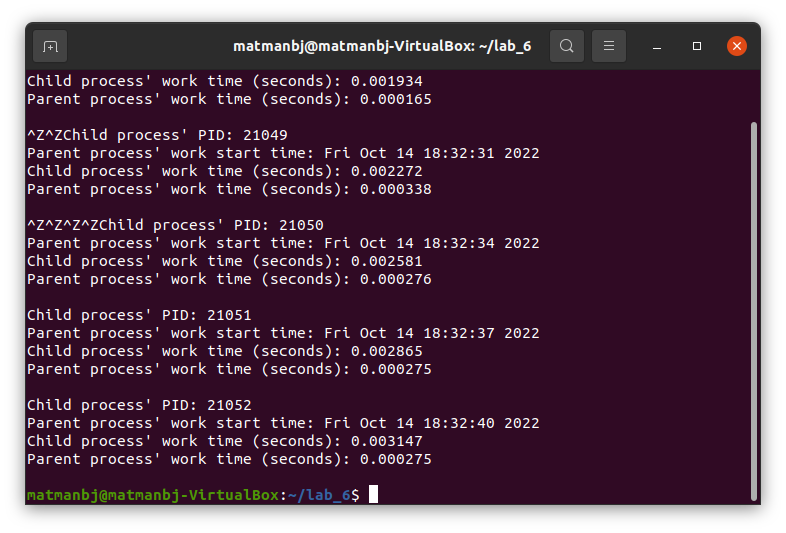


Рисунок 3. Запуск программы с периодом запуска и количеством раз, равными «3 5» соответственно, с нажатием клавиш «Ctrl-Z»

# 4. Вывод

В ходе выполнения лабораторной работы №6 «Организация периодических процессов» были изучены системные функции, которые позволяли периодически запускать программу, была изучена возможность устанавливать пользователем количество запусков и период в секундах, через который эти запуски будут повторяться. Таким образом и было изучено использование сервиса cron, механизма сигналов и интервальных таймеров для организации периодических процессов.

# 5. Список использованных источников

1. Онлайн-курс «Организация процессов и программирование в среде Linux» в LMS Moodle [сайт]. URL: <https://vec.etu.ru/moodle/course/view.php?id=9703>.

2. Разумовский Г.В. Организация процессов и программирование в среде Linux: учебно-методическое пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2018. 40с.