## 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»

**Тема: Организация периодических процессов**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Яловега Н.В. |
| Преподаватель |  | Разумовский Г.В. |

Санкт-Петербург

2022

# Цель работы

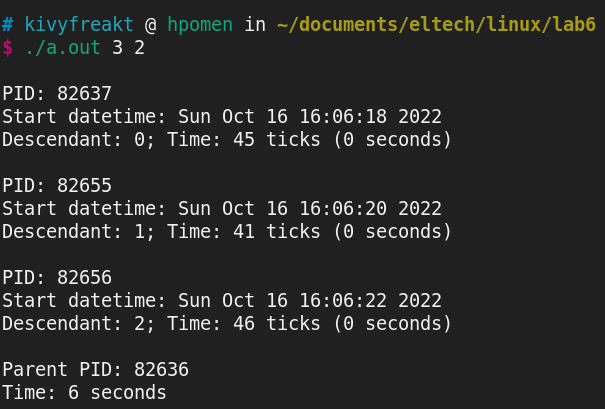
Использование сервиса cron, механизма сигналов и интервальных таймеров для организации периодических процессов.

# Задание

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.

# Примеры выполнения программы

Запустим программу выбрав интервал запуска 2 секунды и количество запусков равное 3-ём.



**Рисунок 1.** Результат выполнения программы.

# Исходный код программ

## main.cpp

# #include <sys/time.h>

# #include <sys/wait.h>

# #include <stdlib.h>

# #include <unistd.h>

# #include <time.h>

# #include <signal.h>

# #include <iostream>

# void signal\_handler(int sig)

# {

# if (sig == SIGALRM)

# if (fork() == 0)

# {

# time\_t start\_time = time(NULL);

# std::cout << "\nPID: " << getpid() << std::endl;

# std::cout << "Start datetime: " << ctime(&start\_time);

# exit(EXIT\_SUCCESS);

# }

# }

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

# {

# if(argc == 3)

# {

# int starts = atoi(argv[1]);

# int period = atoi(argv[2]);

# signal(SIGTSTP, SIG\_IGN);

# signal(SIGALRM, signal\_handler);

# 

# struct itimerval timer\_value;

# timerclear(&timer\_value.it\_interval);

# timerclear(&timer\_value.it\_value);

# timer\_value.it\_interval.tv\_sec = period;

# timer\_value.it\_value.tv\_sec = period;

# setitimer(ITIMER\_REAL, &timer\_value, NULL);

# time\_t pt1, pt2;

# clock\_t pt3, pt4;

# pt1 = time(NULL);

# for(int i = 0; i < starts; i++)

# {

# pause();

# pt3 = clock();

# wait(nullptr);

# pt4 = clock();

# std::cout << "Descendant: " << i;

# std::cout << "; Time: " << pt4-pt3 << " ticks (" << (pt4-pt3)/CLOCKS\_PER\_SEC << " seconds)" << std::endl;

# }

# pt2 = time(NULL);

# std::cout << "\nParent PID: " << getpid() << std::endl;

# std::cout << "Time: " << pt2-pt1 << " seconds" << std::endl;

# }

# else

# {

# std::cout << "Incorrect number of arguments!" << std::endl;

# std::cout << "Example: " << argv[0] << " starts period" << std::endl;

# }

# 

# return 0;

# }

# Вывод

При выполнении лабораторной работы мы ознакомились с механизмом работы периодических процессов.