

МИНОБРАЗОВАНИЯ 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

Преподаватель

Яловега Н.В.

Разумовский Г.В.

Санкт-Петербург

2022

Цель работы

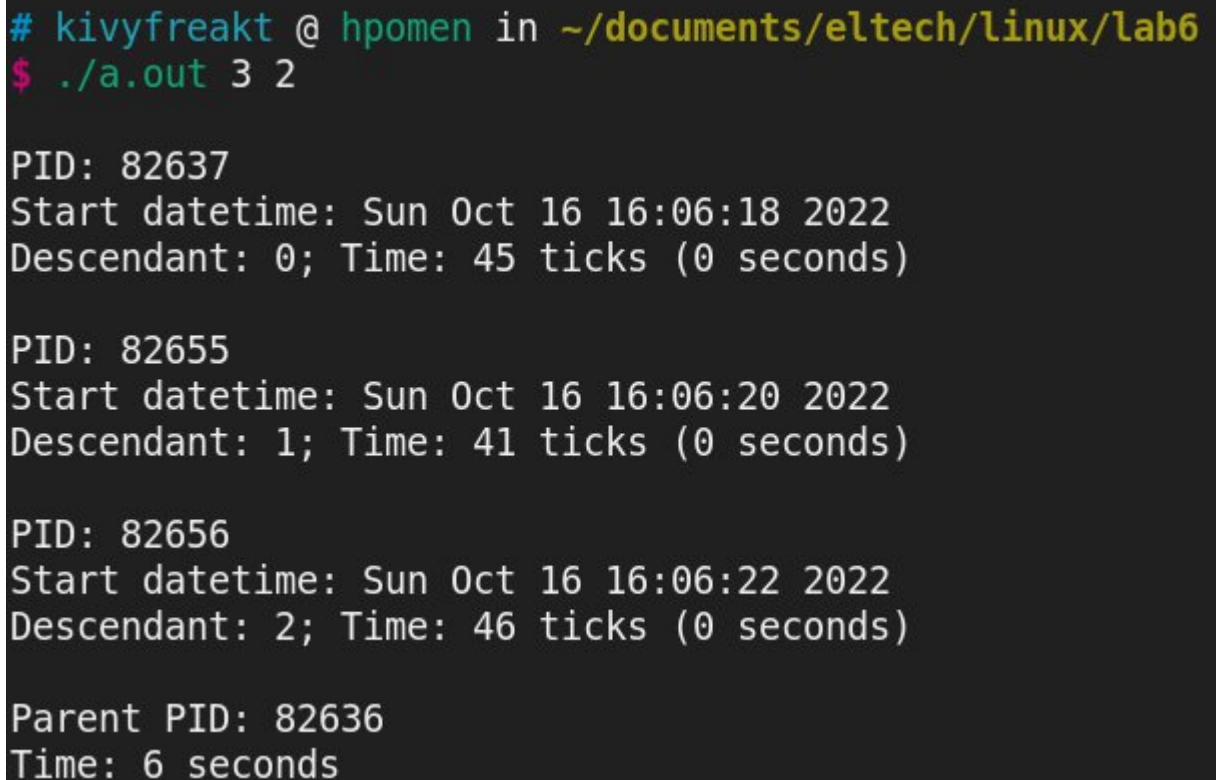
Использование сервиса `cron`, механизма сигналов и интервальных таймеров для организации периодических процессов.

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.



```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab6
$ ./a.out 3 2

PID: 82637
Start datetime: Sun Oct 16 16:06:18 2022
Descendant: 0; Time: 45 ticks (0 seconds)

PID: 82655
Start datetime: Sun Oct 16 16:06:20 2022
Descendant: 1; Time: 41 ticks (0 seconds)

PID: 82656
Start datetime: Sun Oct 16 16:06:22 2022
Descendant: 2; Time: 46 ticks (0 seconds)

Parent PID: 82636
Time: 6 seconds
```

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.