МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный ядерный исследовательский университет «МИФИ»
(НИЯУ МИФИ)
Институт Интеллектуальных Кибернетических Систем
Кафедра Кибернетики

Лабораторная работа №2
По курсу «Разработка программного обеспечения ОС UNIX»

Работу выполнил студент группы Б17-511: Павлов И.Е.

Проверил: Ктитров С.В.

Москва 2020

Постановка задачи

Задание по курсу «Разработка ПО ОС UNIX» **Вариант В–25**

Разработать программу, запускающую задаваемую программу (возможно, с аргументами) и измеряющую время ее выполнения в секундах. По ее окончании следует сообщить, не была ли программа прервана сигналом, если да, то каким. Программу оформить как утилиту командной строки.

Листинг программы

```
#include <unistd.h>
#include <sys/wait.h>
#include <iostream>
#include<errno.h>

// gcc 2.cpp -lstdc++ -o 2
// ./2 /Users/pavlov/Desktop/1 1 /Users/pavlov/Desktop/file.pdf
/Users/pavlov/Desktop/newdir

// !!! указываются аругменты при запуске: путь к файлу, название команды, аргументы
команды !!!

int main(int argc, char* argv[])
{
    struct timespec cur_time;
    clock_gettime(CLOCK_REALTIME, &cur_time);
    int begin_s = cur_time.tv_sec;
    int begin_ns = cur_time.tv_nsec;
    pid_t pid = fork();
    if (pid == 1)
        std::cout << "Ошибочка fork()" << std::endl;
    if (pid == 0) {
        int ret;

//      создаем массив для ЗАПУСКА дочернего процесса
        char *arguments[argc - 1];
        for(int i = 0; i < argc-2; i++)
            arguments[i] = argv[i+2];
        arguments[argc-1] = (char *)0;

        ret = execv(argv[1], arguments);
        perror("EXEC:");
        std::cout << std::endl;
        exit(1);
    }
    else {
        std::cout << "запущен процесс" << std::endl;
        int stat_val;
        pid_t w;
        do {
            w = waitpid(pid, &stat_val, WUNTRACED | WCONTINUED);
            if (w == -1) {
                perror("waitpid");
                exit(EXIT_FAILURE);
            }
            if (WIFEXITED(stat_val)) {
                std::cout << "exited, status = " << WEXITSTATUS(stat_val) <<
std::endl;
            } else if (WIFSIGNALED(stat_val)) {
                std::cout << "killed by signal " << WTERMSIG(stat_val) << std::endl;
            } else if (WIFSTOPPED(stat_val)) {
                std::cout << "stopped by signal " << WSTOPSIG(stat_val) <<
std::endl;
            } else if (WIFCONTINUED(stat_val)) {
                std::cout << "continued" << std::endl;
            }
        } while (!WIFEXITED(stat_val) && !WIFSIGNALED(stat_val));
        clock_gettime(CLOCK_REALTIME, &cur_time);
        std::cout << "время работы программы " << argv[2] << " = " <<
cur_time.tv_sec - begin_s << " секунд " << cur_time.tv_nsec - begin_ns << "
наносекунд" << std::endl;
        return 0;
    }
}
```