

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ  
(ТУСУР)

**Кафедра автоматизированных систем  
управления (АСУ)**

**Отчет по лабораторной работе № 4 по дисциплине  
«Операционные системы»**

Выполнил:

Небиев Н.И.

«13» июня 2023 г.

Проверил: \_\_\_\_\_

\_\_\_\_\_

«\_\_\_\_» \_\_\_\_ 20\_\_ г.

Калининград 2023г  
СОДЕРЖАНИЕ

Цель работы.....	2
Задача.....	2
Выполнение задач.....	2
Вывод.....	6

## Цель работы.

Целью выполнения настоящей лабораторной работы является получение навыков программного управления процессами с помощью сигналов.

## Задача.

Процесс-отец открывает существующий текстовый файл, а затем порождает два дочерних процесса, которые по очереди выводят содержимое этого файла фиксированными порциями по 15 символов, предваряя каждый вывод номером своего процесса. Вывод на экран заканчивается или при достижении середины файла, или по истечении интервала времени в 5 секунд.

## Выполнение задач.

- 1) Создадим файлы, с которыми будем работать.

```
linux-atvp:/home/admin/dirE # touch signals.c test.txt test2.txt
linux-atvp:/home/admin/dirE # ls
signals.c test.txt test2.txt
linux-atvp:/home/admin/dirE #
```

- 2) Файл test.txt будет довольно большим, чтобы программа не успела его обработать до середины и выполнение завершилось через пять секунд.

```

Remarks
The _lseek function moves the file pointer
associated with fd to a new location that is offset bytes from origin.
The next operation on the file occurs at the new location. The origin argument
must be one of the following constants, which are defined in Stdio.h.Return Value
_lseek returns the offset, in bytes, of the new position
from the beginning of the file. _lseeki64 returns the offset
in a 64-bit integer. The function returns -1L to indicate
an error. If passed an invalid parameter, such as a bad
file descriptor, or the value for origin is invalid or the
position specified by offset is before the beginning of the
file, the invalid parameter handler is invoked, as described
in Parameter Validation. If execution is allowed to continue,
these functions set errno to EBADF and return -1L. On devices
incapable of seeking (such as terminals and printers), the return
value is undefined.
For more information about these and other error codes,
see _doserrno, errno, _sys_errlist, and _sys_nerr.
Remarks
The _lseek function moves the file pointer
associated with fd to a new location that is offset bytes from origin.
The next operation on the file occurs at the new location. The origin argument
must be one of the following constants, which are defined in Stdio.h.Return Value
_lseek returns the offset, in bytes, of the new position
from the beginning of the file. _lseeki64 returns the offset
in a 64-bit integer. The function returns -1L to indicate
an error. If passed an invalid parameter, such as a bad
file descriptor, or the value for origin is invalid or the
position specified by offset is before the beginning of the
file, the invalid parameter handler is invoked, as described
in Parameter Validation. If execution is allowed to continue,
these functions set errno to EBADF and return -1L. On devices
incapable of seeking (such as terminals and printers), the return
value is undefined.
For more information about these and other error codes,
see _doserrno, errno, _sys_errlist, and _sys_nerr.linux-atvp:/home/admin/dirE #

```

- 3) Файл test2.txt будет небольшим, чтобы программа не могла выполняться более 5 секунд и обработала только первую половину файла.

```

linux-atvp:/home/admin/dirE # cat test2.txt
test for end out in the middle of file. How do you do? Hello, goodbye!linux-atvp:/home/admin/dirE #

```

- 4) В данном случае создадим только два процесса, хотя это значение может быть и другим.

```

#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include <err.h>
#include <sys/wait.h>
#include <sys/file.h>

#define PROC_COUNT 2

int pid, np;

void sig_handler (int sig)
{
    if (sig == SIGUSR1) if (np == PROC_COUNT) {
        sleep(1);
        kill (pid, SIGUSR1);
    }
}

int main (int ac, char *av[])
{
    int fd = open (av[1], O_RDONLY);
    if (fd == -1) err (1, "Open file");

```

```

        if (signal (SIGUSR1, sig_handler) == SIG_ERR) err (1, "Set Signal
Handler");
        int prev_pid = getpid();

        for (np=0; np < PROC_COUNT; np++) {
            pid = fork();
            if (!pid) break;
            prev_pid = pid;
        }
        if (np == PROC_COUNT) {
            sleep(1);
            kill (pid, SIGUSR1);
            for (int i= 0; i<np; i++) wait(0);
            close (fd);
            return 0;
        }
        time_t endwait;
        time_t start = time(NULL);
        time_t seconds = 5;
        endwait = start + seconds;
        char buf[21];
        for (int i=0; start < endwait; i++) {
            pause();
            int file_size = lseek(fd, 0, SEEK_END);
            int current_offset = i*15*PROC_COUNT + (PROC_COUNT-np-1)*15;
            if (file_size/2 - current_offset <= 0) break;
            if (lseek (fd, current_offset, SEEK_SET) < 0) break;
            int size = read (fd, buf, 15);
            if (size<1) break;
            buf[size]= 0;

            printf ("proc %d: %s\n", getpid(), buf);
            fflush (stdout);

            kill (prev_pid, SIGUSR1);
            start = time(NULL);
        }
        kill (prev_pid, SIGUSR1);
        return 0;
    }
}

```

5) Протестируем программу на большом файле.

```
linux-atvp:/home/admin/dirE # ls
signals.c test.txt test2.txt
linux-atvp:/home/admin/dirE # gcc signals.c
linux-atvp:/home/admin/dirE # ls
a.out signals.c test.txt test2.txt
linux-atvp:/home/admin/dirE # ./a.out test.txt
proc 13434: Return Value
_l
proc 13433: seek returns th
proc 13434: e offset, in by
proc 13433: tes, of the new
proc 13434: position
from
proc 13433: the beginning o
proc 13434: f the file. _ls
proc 13433: eeki64 returns
proc 13434: the offset
in a
proc 13433: 64-bit integer
linux-atvp:/home/admin/dirE # █
```

б) Протестируем программу на небольшом файле.

```
linux-atvp:/home/admin/dirE # ls
a.out signals.c test.txt test2.txt
linux-atvp:/home/admin/dirE # ./a.out test2.txt
proc 13472: test for end ou
proc 13471: tut in the midd
proc 13472: le of file. How
linux-atvp:/home/admin/dirE # █
```

## **Вывод.**

В ходе выполнения данной лабораторной работы были получены навыки программного управления процессами с помощью сигналов. Были изучены основы работы с сигналами, настройка обработчиков сигналов, передача сигналов между процессами. Полученные знания могут быть применены при написании программ, работающих с процессами и обработкой различных событий.