Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работ №2 по курсу**

**«Операционные системы»**

Группа: М80 – 206Б-18
Студент: Макаренкова В.М.
Преподаватель: Соколов А.А.
Оценка: _____
Дата: _____

Москва, 2019.

## Содержание

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

## Постановка задачи.

На вход программе подается название 2-ух неотсортированных файлов. Необходимо отсортировать оба файла (каждый в отдельном процессе) произвольной сортировкой (на усмотрение студента). Из двух отсортированных файлов необходимо получить на выходе один «слитый» отсортированный файл.

## Общие сведения о программе

Программа компилируется из одного файла lab2.c. В данном файле используются заголовочные файлы s#include stdio.h, stdlib.h, unistd.h, fcntl.h, sys/wait.h. В программе используются следующие системные вызовы:

1. **read** – для чтения данных из файла
2. **write** – для записи данных в файл
3. **pipe** – для создания однонаправленного канала, через который могут общаться два процесса. Системный вызов возвращает два дескриптора файлов. Один для чтения из канала, другой для записи в канал.
4. **fork** – для создания дочернего процесса.
5. **wait** – для ожидания завершения дочернего процесса.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Используя системный вызов pipe создать канал, по которому будут обмениваться данными два процесса.
2. Используя системный вызов fork создать дочерний процесс.
3. Пока родительский процесс на записал данные в канал. Дочерний процесс ждет. И как только родительский процесс записал данные в канал дочерний процесс считывает их, производит вычисления и возвращает результат родительскому процессу.
4. Родительский процесс выводит результат используя write.

## Основные файлы программы.

## Файл main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>



#define STR_SIZE 1000
#define BUF_SIZE 1000


int strToInt(int* buf, char* str, int str_size)
{
    int i = 0, buf_size = 0;
    for (i = 0; i < str_size; i++) {
        if (str[i] == ' ' || str[i] == '\n') {
            buf_size++;
        } else if (str[i] == '-') {
            buf[buf_size] = -1 * (str[i+1] - '0');
            i++;
        } else {
            if (buf[buf_size] >= 0) {
                buf[buf_size] = buf[buf_size] * 10 + (str[i] - '0');
            } else {
                buf[buf_size] = buf[buf_size] * 10 - (str[i] - '0');
            }
        }
    }
    return buf_size;
}


void IntTostr(int* buf, int buf_size, char* str)
{
    int i = 0, str_size = 0;
```

```
    for (i = 0; i < STR_SIZE; i++) {
        str[i] = '\0';
    }
    for (i = 0; i < buf_size; i++) {
        str_size += sprintf(str + str_size, "%d", buf[i]);
        str[str_size] = ' ';
        str_size++;
    }
}

void sorti(int*buf, int buf_size)
{
    int ans[buf_size];
    int j;
    int tmp = 0;
    for (int i = 0; i < buf_size; i++) {
        ans[i] = buf[i];
        if (i > 0) {
             j = i - 1;
             while ((ans[j + 1] < ans[j]) && (j >= 0)) {
                 tmp = ans[j + 1];
                 ans[j + 1] = ans[j];
                 ans[j] = tmp;
                 j--;
              }
        }
    }
    for (int i = 0; i <buf_size; i++) {
        buf[i] = ans[i];
    }
}


int main(int argc, char* argv[]) {
    if (argc != 3) {
        printf("Eror. Wrong arguments\n");
        return 1;
    }
```

```
pid_t pid_1 = 0, pid_2 = 0;
int exit_status = 0;

int fd_1[2];
pipe(fd_1);
int fd_2[2];
pipe(fd_2);

char str[STR_SIZE] = {'\0'};
int str_size = 0;
char c = '\0';
int buf[BUF_SIZE] = {0};
int buf_size = 0;

if ((pid_1 = fork()) == 0) {

    close(fd_1[0]); // не понадобится выходной поток
    int file_1 = 0;
    if ((file_1 = open(argv[1], O_RDWR)) == -1) { // O_RDWR — для чтения и записи
        printf("%s: no such file\n", argv[1]);
        exit(1); // выйдет из дочернего процесса
    }

    while (read(file_1, &c, 1)) {
        str[str_size] = c;
        str_size++;
    }

    buf_size = strToInt(buf, str, str_size);
    sorti(buf, buf_size);
    IntTostr(buf, buf_size, str);

    write(fd_1[1], str, sizeof(str));
    close(fd_1[1]);

    exit(0);
}
```

```
    //написать результат если exit_status == 0
    wait(&exit_status);
    if (WEXITSTATUS(exit_status) == 0) { // WEXITSTATUS возвращает код завершения потомка процесса.
        close(fd_1[1]);
        read(fd_1[0], str, sizeof(str));
        printf("%s\n", str);
        close(fd_1[0]);
    }

    if ((pid_2 = fork()) == 0) {

        close(fd_2[0]); // не понадобится выходной поток
        int file_2 = 0;
        if ((file_2 = open(argv[2], O_RDWR)) == -1) {
            printf("%s: no such file!\n", argv[2]);
            exit(1);
        }

        while (read(file_2, &c, 1)) {
            str[str_size] = c;
            str_size++;
        }

        buf_size = strToInt(buf, str, str_size);
        sorti(buf, buf_size);
        IntTostr(buf, buf_size, str);

        write(fd_2[1], str, sizeof(str));
        close(fd_2[1]);

        exit(0);
    }

    //написать результат если exit_status == 0
    wait(&exit_status);
    if (WEXITSTATUS(exit_status) == 0) { // WEXITSTATUS возвращает код завершения потомка.
```

```
        close(fd_2[1]);
        read(fd_2[0], str, sizeof(str));
        printf("%s\n", str);
        close(fd_2[0]);
    }

    return 0;
}
```

## Демонстрация работы программы.

```
prostovera@LAPTOP-UPOTQB9Q:/mnt/c/ucheba/seckurs/os/2$ ./a.out test1.txt test2.txt
0 1 5 6 24 89
-12233445 -2 0 5 80 90 345
prostovera@LAPTOP-UPOTQB9Q:/mnt/c/ucheba/seckurs/os/2$ cat test1.txt
1 5 -0
24 6 89
prostovera@LAPTOP-UPOTQB9Q:/mnt/c/ucheba/seckurs/os/2$ cat test2.txt
345
80
-2
0
-12233445 90 5
```

## Вывод.

Я научился создавать процессы, используя системный вызов fork(), обрел навыки межпроцессного взаимодействия посредством каналов, которые

создаются вызовом pipe(). Узнал что такое вектор прерываний и как работают процессы в ОС Linux. Улучшил навыки программирования на языке программирования Си.