**федеральное государственное бюджетное образовательное учреждение высшего образования «Московский авиационный институт (национальный исследовательский университет)»**Институт “Информационные технологии и прикладная математика”

**Лабораторная работа № 2**

по курсу «Операционные системы»

Тема: Приобретение практических навыков в управлении процессами в ОС, обеспечении обмена данных между процессами посредством каналов

Студент: Зайцев Кирилл

Группа: 80-201

Преподаватель: Зацепин Н.А.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**: Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 7

В файле записаны команды вида: «число число число». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип float. Количество чисел может быть произвольным.

1. Решение:

Для решения задачи использовались следующие системные вызовы:

1. pid\_t fork()

Создание дочернего процесса, если pid = 0, то данный процесс -- ребёнок, если pid > 0, то родитель, если pid < 0, то произошла ошибка.

1. int pipe()

Создание канала для передачи данных между процессами

1. exit()

Закрытие данного процесса

1. int read() -- чтение байтов из файлового дескриптора в буфер
2. int putchar() -- запись символа в стандартный поток вывода

3. Код программы:

**Main.c**

**#include "stdio.h"**

**#include "string.h"**

**#include "unistd.h"**

**#include "sys/wait.h"**

**#include "fcntl.h"**

**#include "stdlib.h"**

**int main(int *argc*, char\* *argv*[]) {**

**char \*name;**

**size\_t len;**

**printf("Enter file:");**

**if(getline(&name, &len, stdin) == -1) {**

**perror("can't get file");**

**}**

**name[strlen(name) - 1] = '\0';**

**int filedescriptor[2];**

**if(pipe(filedescriptor) != 0) {**

**perror("can't create pipe");**

**}**

**int readd = filedescriptor[0];**

**int writed = filedescriptor[1];**

**int file = open(name, O\_RDONLY);**

**if(file == -1) {**

**perror("can't open the file");**

**}**

**pid\_t pid = fork();**

**if(pid == 0) {**

**if(dup2(writed, fileno(stdout)) == -1) {**

**exit(1);**

**}**

**if(dup2(file, fileno(stdin)) == -1) {**

**exit(1);**

**}**

**execl("childprocess", "childprocess", NULL);**

**perror("can't execute child process");**

**exit(EXIT\_FAILURE);**

**} else if(pid == -1) {**

**perror("pid error");**

**}**

**char c;**

**while(read(readd, &c, sizeof(char)) > 0) {**

**putchar(c);**

**}**

**return 0;**

**}**

**Child.c**

**#include <stdio.h>**

**int main(){**

**double a;**

**char c;**

**double res = 0;**

**while(scanf("%lf%c", &a, &c) != EOF) {**

**res += a;**

**if(c == '\n') {**

**printf("%lf\n", res);**

**fflush(stdout);**

**res = 0.;**

**continue;**

**}**

**}**

**printf("%lf\n", res);**

**return 0;**

**}**

4. Тестирование.

kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ cat Test

1.0 2.5

2.5 2.5

3.45 0.55

kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ gcc main.c -o main

kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ ./main

Enter file:Test

12.500000  
 kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ cat Test2

1 2 3

4 5 6 7

8 9 10 11

^C

kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ ./main

Enter file:Test2

66.000000

kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ ./main

Enter file:Test2

66.000000

^C

kirillpc@TuringsMachine:/mnt/e/lab\_repos/OS/lab2$ ./main

Enter file:Test3

19.584670

5. Выводы.

В ходе лабораторной работы я научился работать с каналами передачи данных для межпроцессорного взаимодействия, перенаправлять потоки ввода и вывода, а также работать с файловыми дескрипторами.