Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Студент: Кажекин Денис Андреевич

Группа: М8О-207Б-21

Вариант: 2

Преподаватель: Черемисинов Максим

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/DKazhekin/OS

**Постановка задачи**

**Цель работы**

Целью работы является приобретение практических навыков в управлении процессами в ОС и обеспечение обмена данными между процессами посредством каналов.

**Задание**

Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип float. Количество чисел может быть произвольным.

**Общие сведения о программе**

Программа компилируется из файла parent.c. А также с помощью execvp() вызывается программа child.c вполследствии.

В программе используются следующие системные вызовы:

1. pipe() – создает канал между двумя процессами и возвращает два дескриптора файла.
2. execvp() – функция создает и выполняет новый порожденный процесс, заменяя в памяти родительский процесс на порожденный.
3. mkfifo() – предоставляет процессу именованную трубу в виде объекта файловой системы для передачи данных.
4. fopen() – открывает файл, имя которого указано аргументом.
5. fclose() – закрывает файл, имя которого указано аргументом.

**Общий метод и алгоритм решения**

В родительском процессе мы считываем пользовательское название файла для записи конечного результата и N чисел, с которыми необходимо работать в дочернем процессе. С помощью pipe’а мы передаем длину названия пользовательского файла и само название в дочерний процесс, а также используем FIFO файл для передачи float’овского массива в программу child.c. В дочернем процессе мы принимаем строку названия пользовательского файла и вызываем программу child.c с помощью execvp() передавая аргументом строку названия пользовательского файла. В child.c мы принимаем float’овский массив и суммируем все числа в переменную “sum”, выполняя поставленную задачу, после чего записываем результат в новосозданный файл.

**Исходный код**

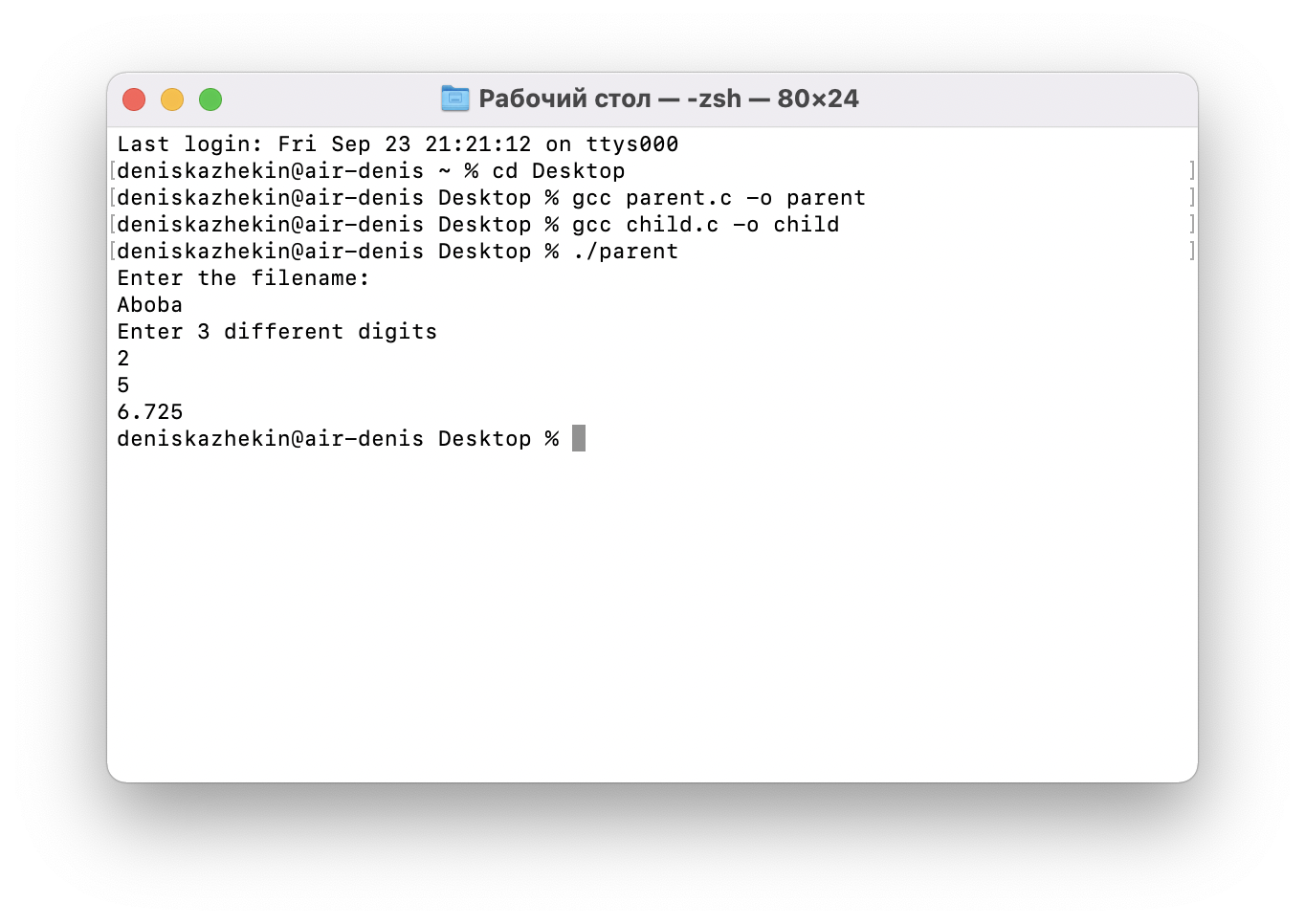
parent.c:

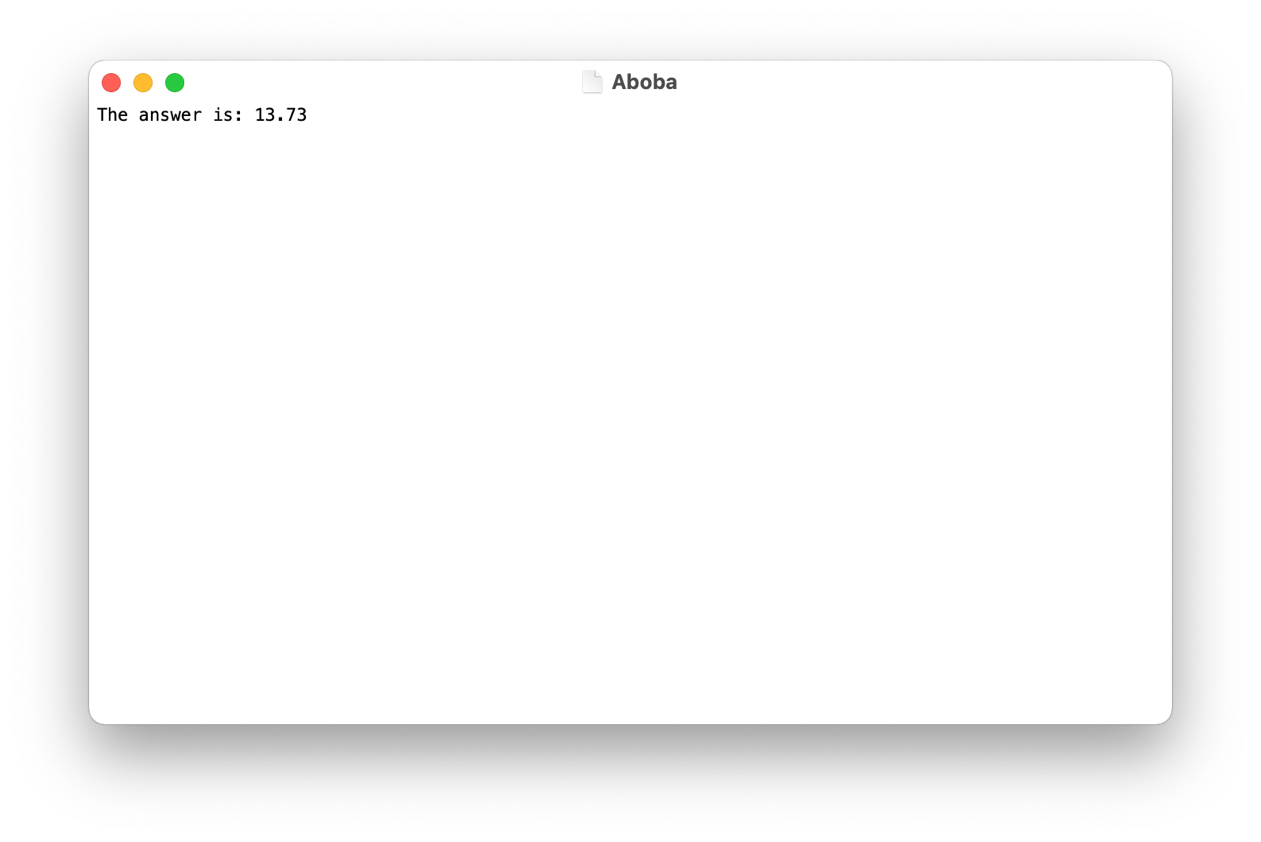
#include "unistd.h"  
#include "stdio.h"  
#include <stdlib.h>  
  
#include "string.h"  
  
#include "sys/stat.h"  
#include "sys/types.h"  
#include "errno.h"  
#include "fcntl.h"  
  
  
unsigned int time(void \*pVoid);  
  
int main(){  
 int fd1[2];  
 if (pipe(fd1) == -1){  
 printf("An error ocurred with opening the pipe\n");  
 }  
  
  
 int id = fork();  
  
 if (id < 0){  
 perror("An error occurred with fork");  
 return -1;  
 }  
  
 else if (id > 0){  
 char s[20];  
 printf("Enter the filename:\n");  
 scanf("%s",s);  
 int len = strlen(s);  
 close(fd1[0]);  
 write(fd1[1], &len, sizeof(int));  
 write(fd1[1], s, sizeof(char) \* len);  
 close(fd1[1]);  
  
 if (mkfifo("myfifo", 0777) == -1) {  
 if (errno != EEXIST) {  
 printf("Couldnt create fifo file");  
 return 1;  
 }  
 }  
  
 int fd = open("myfifo", O\_WRONLY);  
  
 srand(time(NULL));  
  
 int n = rand() % 10 + 1;  
  
 if(write(fd, &n, sizeof(int)) == -1){  
 return 2;  
 }  
  
  
 printf("Enter %d different digits\n", n);  
  
 float data[11];  
 for(int i = 0; i < n; i++){  
 scanf("%f", &data[i]);  
 if(write(fd, &data[i], sizeof(float)) == -1){  
 return 3;  
 }  
 }  
 close(fd);  
 }  
 else{  
 int len;  
 char s[20];  
  
 read(fd1[0], &len, sizeof(int));  
 read(fd1[0], s, sizeof(char) \* len);  
 close(fd1[1]);  
 close(fd1[1]);  
  
 char \*args[] = {"./child", s, NULL};  
 execvp(args[0], args);  
 }  
 return 0;  
}

child.c:

#include "unistd.h"  
#include "stdio.h"  
#include <stdlib.h>  
  
#include "string.h"  
#include "sys/stat.h"  
#include "sys/types.h"  
#include "errno.h"  
#include "fcntl.h"  
  
  
int main(int argc, char\* argv[]){  
 int n;  
 float arr[11];  
   
 int fd = open("myfifo", O\_RDONLY);  
   
 if (fd == -1){  
 return 1;  
 }  
   
 if(read(fd, &n, sizeof(int)) == -1){  
 return 2;  
 };  
   
 for(int i = 0; i < n; i++){  
 if (read(fd, &arr[i], sizeof(float)) == -1){  
 return 3;  
 }  
 }  
   
 float sum = 0;  
   
 for(int i = 0; i < n; i++){  
 sum += arr[i];  
 }  
   
 FILE \*fptr;  
 fptr = fopen(argv[1],"w");  
   
 fprintf(fptr,"The answer is: %.2f\n", sum);  
 fclose(fptr);  
   
 close(fd);  
 return 0;  
}

**Демонстрация работы программы**





**Выводы**

Я приобрел практические навыки в управлении процессами в ОС и обеспечение обмена данными между процессами посредством каналов, успешно выполнив лабораторную работу №2