Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа № 4 по курсу

«Операционные системы»

Студент: Румынина Екатерина Александровна
Группа: М8О-201Б-21
Вариант: 7
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/KetRum0/mai_os_labs

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 через отображаемые файлы (memory-mapped files).
Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.
Вариант аналогичен лабораторной работе 2.

Общие сведения о программе

main.c - основная программа, которая считывает ввод и перенаправляет его в

родительский процесс

parent.c – программа для реализации родительского процесса

child.c – программа для реализации дочернего процесса

parent.h – заголовочный файл с объявлением ParentRoutine

Общий метод и алгоритм решения

Аналогичен лабораторной работе 2, но вместо передачи через pipe используется передача

через отображаемый файл и семафор для синхронизации процессов.

Исходный код

main.c

```
#include "parent.h"
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    char name[256];
    scanf("%s",name);
    float result;
    result = ParentRoutine(name);
    printf("%g\n",result);
    return 0;
}
```

parent.h

```
#ifndef OS_LABS_PARENT_H
#define OS_LABS_PARENT_H
#include <stdio.h>
float ParentRoutine(char *filename);
#endif //OS_LABS_PARENT_H
```

parent.c

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>

float ParentRoutine(char *filename){
                              // char name[256];
                              // scanf("%s",name);
                              int file;
                              if ((file = open(filename, O_RDONLY))==-1){
                                    perror("open error");
                                    exit(EXIT_FAILURE);
                              }
                              sem_t *sem = sem_open("my_sem", O_CREAT,S_IRUSR
| S_IWUSR, 0);
    if (sem == SEM_FAILED){
                                    perror("sem_open error");
        exit(EXIT_FAILURE);
    }
                              int id = fork();
                              if (id == -1){
                                    perror("fork error");
                                    exit(EXIT_FAILURE);

                              } else if (id == 0){ //child
        dup2(file,STDIN_FILENO);
```

```
                                    close(file);
                                    if (execlp("./child", "./child", NULL) ==
-1){
                                          perror("execlp error");
                                          exit(EXIT_FAILURE);
                                    }

    } else { //parent
                                    int fd = shm_open("shared_file", O_RDWR |
O_CREAT, S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH);
                                    if (fd == -1){
                                          perror("shm_open error (parent)");
                                          exit(EXIT_FAILURE);
                                    }
        sem_wait(sem);
                                    sem_close(sem);
        struct stat s;
        fstat(fd, &s);
        int map_size = s.st_size;
        char *mapped = (char *) mmap(NULL, map_size, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
                                    if (mapped == MAP_FAILED) {
            perror("mmap error (parent)");
                              exit(EXIT_FAILURE);
        }
                                    close(fd);
                                    float result = atof(mapped);
                                    if (munmap(mapped,map_size) == -1){
                                          perror("munmap error (parent)");
                                          exit(EXIT_FAILURE);
                                    }
                                    if (shm_unlink("shared_file") == -1){
                                          perror("shm_unlink error");
                                          exit(EXIT_FAILURE);
        }

                                    return result;
                              }
```

```
}
```

child.c

```
#include <unistd.h>
#include <stdio.h>
#include <semaphore.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
int main(int argc, char *argv[]){
    sem_t *sem = sem_open("my_sem", 1);
    if (sem == SEM_FAILED){
                                perror("sem_open error (child)");
        exit(EXIT_FAILURE);
    }
    float res = 0;
    float x;
    while (scanf("%f",&x)!=EOF){
        res+=x;
    }

    int maxDigits = 256;
    int digits = 0;
    char res_arr[maxDigits];
    gcvt(res, maxDigits, res_arr);
    // int flag = 0;
    // for (int i = 0; i < maxDigits; i++) {
    //     if (res_arr[i]=='.') flag=1;
    //     if (res_arr[i]=='0' & flag==1){
    //         break;
    //     }
    //     digits++;
    // }
    int map_size = maxDigits;
    int fd = shm_open("shared_file", O_RDWR, 0777);
```

```
    if (ftruncate(fd, map_size) == -1){
        perror("ftruncate (child)");
                                exit(EXIT_FAILURE);
    }
    if (fd == -1){
                                perror("shm_open error (child)");
                                exit(EXIT_FAILURE);
    }
    char *mapped = (char *)mmap(NULL, map_size, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
    close(fd);
    if (mapped == MAP_FAILED){
                                perror("mmap error (child)");
        exit(EXIT_FAILURE);
    }
    for (int i = 0; i < map_size; i++) {
        mapped[i] = res_arr[i];
    }
    if (munmap(mapped,map_size) == -1){
                                  perror("munmap error (child)");
                                  exit(EXIT_FAILURE);
                              }
    sem_post(sem);
    sem_close(sem);
}
```

Демонстрация работы программы

```
ket@ket-laptop:~/Desktop/mai_os_labs/lab4$ cat test.txt
1 2 3 4 -1 0.1 0.11
ket@ket-laptop:~/Desktop/mai_os_labs/lab4$ ./lab4
test.txt
9.21
```

Выводы

В результате выполнения данной лабораторной работы я освоила принципы работы с файловыми системами и научилась обеспечивать обмен данных между процессами посредством технологии «File mapping»