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: Люгге Т.В.

Группа: М8О-201Б-21

Вариант: 2

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**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/MonkeDLyugge/LabOS

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в:

1. Освоение принципов работы с файловыми системами
2. Обеспечение обмена данных между процессами посредством технологии «File mapping»

**Задание**

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.

**Общие сведения о программе**

В программе используются следующие системные вызовы:

1. unlink() – удаление имени из файловой системы
2. fork() – создание дочернего процесса
3. open() – открытие файла
4. close() – закрытие файла
5. write() – запись последовательности байт
6. mmap() - создание отражения файла в памяти
7. munmap() - удаление отражения файла в памяти

**Исходный код**

|  |
| --- |
| **parent.cpp** |
| // Parent process would provide data collecting and transmission to the child process  #include <cstdlib>  #include <iostream>  #include "unistd.h"  #include "stdio.h"  #include "stdlib.h"  #include "sys/mman.h"  #include "string.h"  #include "errno.h"  #include "semaphore.h"  #include "signal.h"  #include "fcntl.h"  #include "pthread.h"  #include "../include/parent.hpp"  int ParentProcess(FILE\* standartInput) {  // The entry point to the parent process  const int SIZE = sizeof(float);  unlink("file1");  unlink("file2");  int file1 = open("file1", O\_RDWR | O\_CREAT, S\_IRUSR | S\_IWUSR);  int file2 = open("file2", O\_RDWR | O\_CREAT, S\_IRUSR | S\_IWUSR);  if (file1 == -1 || file2 == -1) {  perror("file open error");  return EXIT\_FAILURE;  }  if (ftruncate(file1, SIZE) == -1) {  perror("ftruncate error");  return EXIT\_FAILURE;  }  if (ftruncate(file2, SIZE) == -1) {  perror("ftruncate error");  return EXIT\_FAILURE;  }  sem\_t\* sem1 = sem\_open("semaphore1", O\_CREAT, S\_IRUSR | S\_IWUSR, 0);  sem\_t\* sem2 = sem\_open("semaphore2", O\_CREAT, S\_IRUSR | S\_IWUSR, 0);  if (sem1 == SEM\_FAILED || sem2 == SEM\_FAILED) {  perror("sem open error");  return EXIT\_FAILURE;  }  int pid = fork();  if (pid == -1) {  // Fork error  return -1;  }  if (pid != 0) {  // Opening pipe  void\* out = mmap(NULL, SIZE, PROT\_WRITE, MAP\_SHARED, file1, 0);  void\* ans = mmap(NULL, SIZE, PROT\_READ, MAP\_SHARED, file2, 0);  if (out == MAP\_FAILED || ans == MAP\_FAILED) {  perror("mmap error");  return EXIT\_FAILURE;  }  float num;  while (std::cin >> num) {  memcpy(out, &num, sizeof(float));  sem\_post(sem1);  sem\_wait(sem2);  }  kill(pid, SIGKILL);  munmap(out, SIZE);  munmap(ans, SIZE);  sem\_close(sem1);  sem\_close(sem2);  close(file1);  close(file2);  unlink("file1");  unlink("file2");  } else {  void\* in = mmap(NULL, SIZE, PROT\_READ, MAP\_SHARED, file1, 0);  void\* ans = mmap(NULL, SIZE, PROT\_WRITE, MAP\_SHARED, file2, 0);  sem\_t\* sem1 = sem\_open("semaphore1", O\_CREAT, S\_IRUSR | S\_IWUSR, 0);  sem\_t\* sem2 = sem\_open("semaphore2", O\_CREAT, S\_IRUSR | S\_IWUSR, 0);  if (sem1 == SEM\_FAILED || sem2 == SEM\_FAILED) {  perror("sem open error");  return EXIT\_FAILURE;  }  if (in == MAP\_FAILED || ans == MAP\_FAILED) {  perror("mmap error");  return EXIT\_FAILURE;  }  unlink("result.txt");  int fout = open("result.txt", O\_CREAT | O\_WRONLY, S\_IRUSR);  if (fout == -1) {  perror("open error");  return EXIT\_FAILURE;  }  // // if (dup2(fout, 1) == -1) {  // perror("dup2 error");  // return EXIT\_FAILURE;  // }  float sum = 0;    float num;  sem\_wait(sem1);  memcpy(&num, in, sizeof(float));  sem\_post(sem2);  while (num) {  sum += num;  sem\_wait(sem1);  memcpy(&num, in, sizeof(float));  sem\_post(sem2);  }  sem\_post(sem2);  printf("%.3f", sum);  }  return EXIT\_SUCCESS;  } |

**parent.hpp**

// Parent header file

#ifndef PARENT\_HPP

#define PARENT\_HPP

#include <stdio.h>

#include <iostream>

int ParentProcess(FILE\* standartInput);

#endif**​**

**Демонстрация работы программы**

[microhacker@microhacker-HLYL-WXX9](mailto:microhacker@microhacker-HLYL-WXX9):~/Desktop/LabOS$ ./parent.out

1 2 3

0.5 0.7

0.2

[microhacker@microhacker-HLYL-WXX9](mailto:microhacker@microhacker-HLYL-WXX9):~/Desktop/LabOS$ cat result.txt

7.400

**Выводы**

Составлена и отлажена программа на языке Си, осуществляющая работу и взаимодействие между процессами с использованием отображаемых файлов. Так, получены навыки в обеспечении обмена данных между процессами посредством технологии «File mapping».