МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Лабораторная работа №3

По курсу «Операционные системы»

Студент: Степанов Н.Е.
Группа: М8О-208Б-23
Вариант: 2
Преподаватель: Миронов Е. С.

Дата: _____

Оценка: _____

Подпись: _____

Москва, 2024

Содержание

Репозиторий

https://github.com/n0w3e/os_labs/tree/lab3

Постановка задачи

Цель работы

Приобретение практических навыков в:

Освоение принципов работы с файловыми системами

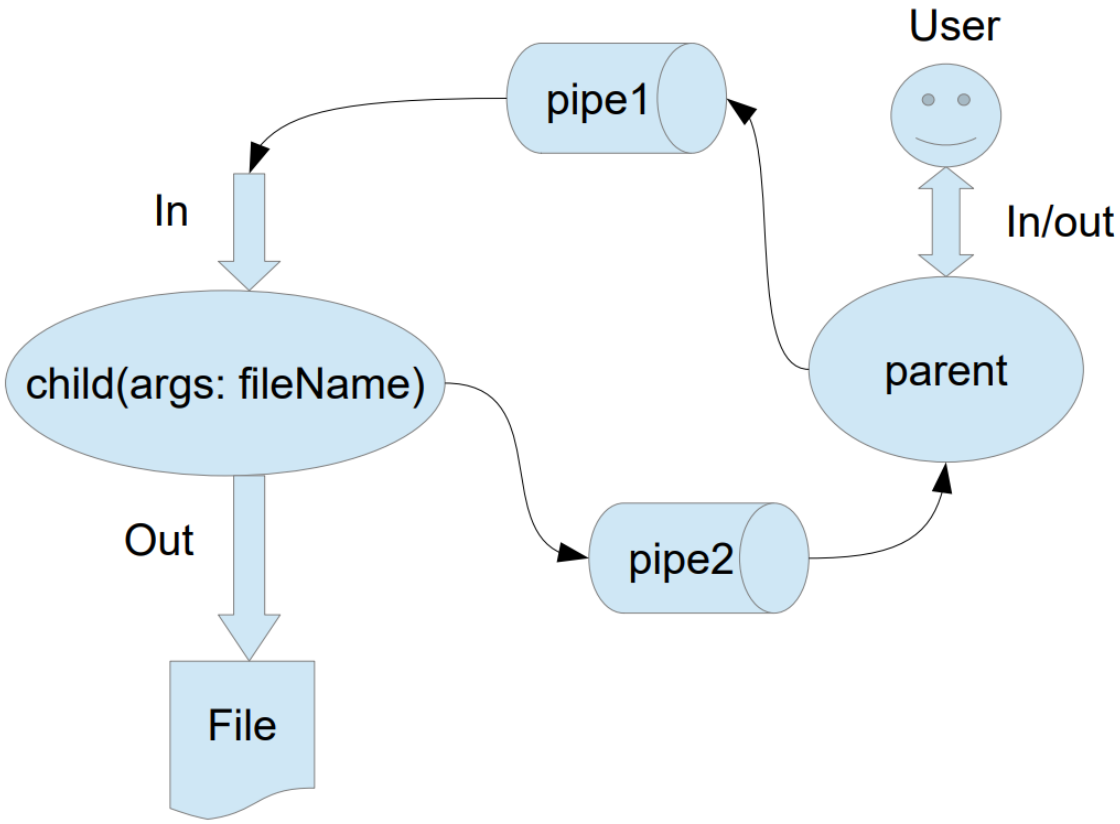
Обеспечение обмена данных между процессами посредством технологии «File mapping»

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.

Вариант задания:

Группа вариантов 1



2 вариант) Пользователь вводит команды вида: «число число число». 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 использует стандартные библиотеки iostream, fstream, sstream для работы с потоками ввода и вывода. В программе используются следующие системные вызовы:

1. fork() - создание дочернего процесса.

2. execl() - замена текущего процесса на выполнение другой программы.

3. waitpid() - ожидание завершения дочернего процесса.

4. mmap() - отображение файла в память для эффективного доступа.

5. open(), close(), write(), read() - работа с файлами.

6. fstat() - получение информации о файле.

Общий метод и алгоритм решения

Родительский процесс отвечает за подготовку данных и запуск дочернего процесса. Дочерний процесс выполняет вычисления (например, суммирование чисел) и записывает результат. Читается содержимое файла. Выполняются вычисления. Результат записывается в файл.

Если аргументов нет (**argc == 1**), вызывается **runParentProcess("data.txt")**. Если передан один аргумент (**argc == 2**), вызывается **runChildProcess(argv[1])**. Создается файл с тем же именем, что и входной файл, для записи данных. Данные из входного файла записываются в созданный файл. Используется **fork()** для создания дочернего процесса. Если **fork()** завершился успешно, дочерний процесс заменяется на выполнение текущей программы с помощью **execl()**. Родительский процесс ожидает завершения дочернего процесса с помощью **waitpid()**.

Открывается файл, переданный родительским процессом. Если файл не найден, выводится сообщение об ошибке. Используется **mmap()** для отображения содержимого файла в память. Содержимое файла (числа) считывается из памяти. Выполняется суммирование чисел. Результат (сумма чисел) записывается в файл **result.txt**. Освобождается отображение памяти с помощью **munmap()**. Закрывается файл с помощью **close()**.

Родительский процесс завершает работу после ожидания завершения дочернего процесса. Дочерний процесс завершает работу после записи результата.

Исходный код

child.h:

```
#ifndef CHILD_H
#define CHILD_H
#include <string>
#include <vector>

void runChildProcess(const std::string& filename);

#endif
```

parent.h:

```
#ifndef PARENT_H
#define PARENT_H

#include <string>
#include <vector>

void runParentProcess(const std::string& filename);

#endif
```

child.cpp:

```
#include "child.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
```

```
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

void runChildProcess(const std::string& filename) {
    int fd = open(filename.c_str(), O_RDONLY);
    if (fd == -1) {
        perror("Ошибка открытия файла");
        return;
    }

    struct stat sb;
    if (fstat(fd, &sb) == -1) {
        perror("Ошибка получения информации о файле");
        close(fd);
        return;
    }

    char* data = static_cast<char*>(mmap(nullptr, sb.st_size, PROT_READ, MAP_PRIVATE, fd, 0));
    if (data == MAP_FAILED) {
        perror("Ошибка отображения файла");
        close(fd);
        return;
    }

    std::istringstream iss(std::string(data, sb.st_size));
    float num, sum = 0;
```

```
    while (iss >> num) {
        sum += num;
    }

    std::ofstream outFile("result.txt");
    if (!outFile) {
        std::cerr << "Ошибка открытия файла для записи результата\n";
    } else {
        outFile << "Сумма чисел: " << sum << std::endl;
        outFile.close();
    }

    munmap(data, sb.st_size);
    close(fd);
}
```

parent.cpp:

```
#include "parent.h"
#include <iostream>
#include <fstream>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>

void runParentProcess(const std::string& filename) {
    std::ifstream inputFile(filename);
    if (!inputFile.is_open()) {
        std::cerr << "Ошибка: входной файл не найден: " << filename << std::endl;
```

```
    return;
}

std::string input((std::istreambuf_iterator<char>(inputFile)),
                  std::istreambuf_iterator<char>());
inputFile.close();

int fd = open(filename.c_str(), O_RDWR | O_CREAT | O_TRUNC, 0666);
if (fd == -1) {
    perror("Ошибка открытия файла");
    return;
}

if (write(fd, input.c_str(), input.size()) == -1) {
    perror("Ошибка записи в файл");
    close(fd);
    return;
}
close(fd);

pid_t pid = fork();

if (pid == -1) {
    perror("Ошибка создания дочернего процесса");
    return;
}

if (pid == 0) {
    execl("./lab3", "./lab3", filename.c_str(), nullptr);
```

```
        perror("Ошибка вызова execl");
        exit(EXIT_FAILURE);
    } else {
        int status;
        waitpid(pid, &status, 0);
    }
}
```

main.cpp:

```
#include "parent.h"
#include "child.h"
#include <iostream>
#include <string>

int main(int argc, char* argv[]) {
    if (argc == 1) {
        runParentProcess("data.txt");
    } else if (argc == 2) {
        runChildProcess(argv[1]);
    } else {
        std::cerr << "Некорректные аргументы\n";
        return 1;
    }
    return 0;
}
```

Демонстрация работы программы

n0wee@DESKTOP-8QSPN1P:~/Coding/os_labs/build/lab3$ ./lab3

Введите числа через пробел: 10 20 30 40 50

1 2 3 4 5

3 4 5

Содержимое файла result.txt:

Сумма чисел: 150

Сумма чисел: 15

Сумма чисел: 12

Выводы

В ходе выполнения данной лабораторной работы я изучил основы работы с процессами в Unix-подобных системах. Я научился работать с файлами через отображение в память с помощью **mmap()**, что оказалось эффективным способом обработки больших объемов данных. Работа с процессами и файлами стала полезным опытом, который поможет мне в изучении системного программирования.