Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

Тема работы

Освоение принципов работы с файловыми системами

Обеспечение обмена данных между процессами посредством технологии «File mapping»

Студент: Шаларь Игорь Павлович
Группа: М8О-208Б-20
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/IgShalar/OS/tree/main/os_lab4

Постановка задачи

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский процесс передает команды пользователя дочернему процессу. Пользователь вводит команды вида: «число<endline>». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются. Взаимодействие между процессами осуществляется через отображаемые файлы.

Общие сведения о программе

Запуск:

_$ cmake .

_$ make

_$ ./parent

Написано для Unix.

parent.cpp: получает название файла, создаёт дочерний процесс и

отображаемый файл.

Системные вызовы:

fork() - создаёт дочерний процесс.

shm_open() - создает или открывает файл.

ftruncate() - устанавливают длину файла.

mmap() - создает новое отображение в виртуальном адресном пространстве вызывающего процесса.

munmap() - удаляет все отражения из заданной области памяти.

execl()- заменяет текущий образ процесса новым образом процесса: выполняет child.cpp и выделяет ему другую область памяти(память parent становится недоступна).

child.cpp: получает число и название файла. Создаёт файл и выполняет проверку на простоту.

Системные вызовы:

_.open() - создаёт или открывает файл.

shm_open()

mmap()

usleep() - ожидание.

shm_unlink() - функция удаляет имя объекта общей памяти.

Общий метод и алгоритм решения

Проверка на простоту:

Проверяю делители ли все числа, большие 1, до корня числа включительно.

Исходный код

parent.cpp:

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

using namespace std;

const int wait_time = 100000;

int main(){
   string t;
```

```
    const char * s;
    getline(cin, t);
    s = t.c_str();
    int a;
    cin >> a;
    int fd = shm_open("file1", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
    if (fd == -1){
        perror("shm_open error");
        return -1;
    }
    if (ftruncate(fd, 2 * sizeof (int)) == -1){
        perror("ftruncate error");
        return -1;
    }
    int * ptr = (int *) mmap(NULL, 2 * sizeof (int), PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
    if (ptr == MAP_FAILED){
        perror("mmap error");
        return -1;
    }
    int id = fork();
    switch (id){
        case -1: {
            perror("fork error");
            return -1;
        }
        case 0: {
            if (execl("b.out", s, NULL) < 0){
                perror("execl error");
                return -1;
            }
            break;

        }

        default: {
            * ptr = a;
            * (ptr + 1) = 1;
            if (munmap(ptr, 2 * sizeof (int)) == -1){
                perror("munmap error");
                return -1;
            }

        }
    }
    return 0;
}
```

child.cpp:

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

using namespace std;

const int wait_time = 100000;

int main(int a, char * s[]){
    int fd = shm_open("file1", O_RDONLY, S_IRUSR);
    if (fd == -1){
        perror("shm_open error");
        return -1;
    }
    int * ptr = (int *) mmap(NULL, 2 * sizeof (int), PROT_READ, MAP_SHARED, fd, 0);
    if (ptr == MAP_FAILED){
        perror("mmap error");
        return -1;
    }
    ofstream f;
    f.open(* s);
    if (!f){
        perror("Can`t create file");
        return -1;
    }
    int b, c = 2;
    while (* (ptr + 1) != 1) usleep(wait_time);
    b = * ptr;
    while (c * c <= b){
        if (b % c == 0){
            f << b;
            f.close();
            break;
        }
        c++;
    }
    if (munmap(ptr, 2 * sizeof (int)) == -1){
        perror("munmap error");
        return -1;
    }
    if (shm_unlink("file1") == -1){
        perror("shm_unlink error");
        return -1;
    }
    f.close();
    return 0;
```

}

Демонстрация работы программы

1)Ввод:

test.txt
20

test.txt:
20

2)Ввод:
test.txt
3

test.txt:
(пусто)

3)Ввод:
test.txt
-1

test.txt:
(пусто)

Выводы

Получил навыки работы с отображаемыми файлами (memory-mapped files). Если pipe’ ы обычно используются для передачи небольших сообщений между двумя процессами, то memory-mapped files удобно использовать для больших объемов данных и с несколькими процессами.