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: Попов А. Д.

Группа: М8О-208Б-23

Преподаватель: Живалев Е. А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Тема:** Работа с файловыми системами и технологиями отображения памяти в ОС

**Цель работы:** Приобретение практических навыков в:

* Освоении принципов работы с файловыми системами.
* Обеспечении обмена данных между процессами посредством технологии "File mapping" (отображаемые файлы).

**Вариант:** 7. В файле записаны команды вида: «число число число<endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип float. Количество чисел может быть произвольным.

**Задачи:**

1. Разработать программу на языке Си, реализующую работу с процессами и их взаимодействие через отображаемые файлы (memory-mapped files).
2. Обеспечить взаимодействие между процессами с использованием семафоров.
3. Обрабатывать системные ошибки, которые могут возникнуть в результате работы программы.
4. Реализовать подсчет суммы чисел в дочернем процессе и вывод результата в родительский процесс.

**Описание решения:**

Программное решение состоит из трех основных компонентов:

1. **Родительский процесс (Parent):**
   * Инициализирует именованные семафоры и отображаемую память.
   * Создает дочерний процесс.
   * Передает данные из файла в отображаемую память.
   * Использует семафоры для управления доступом к памяти.
2. **Дочерние процессы (Child):**
   * Получают доступ к отображаемой памяти и семафорам.
   * Читает строки из отображаемой памяти, вычисляет сумму чисел в каждой строке и записывает результат обратно в отображаемую память.
   * Завершают выполнение при получении специального сигнала "stop".
3. **Вспомогательные модули (utils.c):**
   * Содержат реализацию вспомогательных функций.

**Логика работы программы:**

1. Родительский процесс создает именованные семафоры и отображаемую память с помощью shm\_open и mmap.
2. Пользователь вводит имя файла, который содержит строки с числами.
3. Родительский процесс создает дочерний процесс с помощью fork и перенаправляет стандартный поток ввода дочернего процесса на файл, а стандартный поток вывода — на отображаемую память.
4. Родительский процесс сигнализирует дочернему процессу через семафор о доступности данных.
5. Дочерний процесс читает строки из отображаемой памяти, вычисляет сумму чисел в каждой строке и записывает результат обратно в отображаемую память.
6. Программа завершает работу при вводе специального символа "stop".

**Репозиторий:** https://github.com/aldpopov/OS\_labs/tree/master/LW3

**Исходный код:** Программное обеспечение состоит из следующих файлов:

1. **main.c:** Основная функция, которая вызывает родительский процесс.
2. **parent.c:** Логика работы родительского процесса (инициализация ресурсов, управление дочерним процессом).
3. **child.c:** Логика работы дочернего процесса (обработка строк и запись в отображаемую память).
4. **utils.c:** Реализация вспомогательных функций.

**Пример функции ReverseString:**

// Функция обработки строк в дочернем процессе

int main() {

const int BUFFER\_SIZE = 1024 \* 10;

char buffer[BUFFER\_SIZE];

sem\_t \*semaphore\_write = sem\_open("/semaphore\_write", 0);

sem\_t \*semaphore\_read = sem\_open("/semaphore\_read", 0);

char \*ptr = (char\*) mmap(0, BUFFER\_SIZE, PROT\_WRITE, MAP\_SHARED, STDOUT\_FILENO, 0);

char \*token;

float num;

float sum;

while (fgets(buffer, sizeof(buffer), stdin) != NULL) {

char \*line = strtok(buffer, "\n");

while (line != NULL) {

sem\_wait(semaphore\_write);

sum = 0;

token = strtok(line, " ");

while (token != NULL) {

num = atof(token);

sum += num;

token = strtok(NULL, " ");

}

sprintf(ptr, "%.2f", sum);

sem\_post(semaphore\_read);

line = strtok(NULL, "\n");

}

}

sem\_wait(semaphore\_write);

sprintf(ptr, "%s", "stop");

sem\_post(semaphore\_read);

munmap(ptr, BUFFER\_SIZE);

sem\_close(semaphore\_read);

sem\_close(semaphore\_write);

exit(EXIT\_SUCCESS);

}

**Пример работы:**make run

Enter file's name:

test.txt

Sum: 10.00

Sum: 15.00

Sum: 20.00

**Вывод:** В ходе выполнения лабораторной работы были выполнены все поставленные задачи. Программа успешно организует взаимодействие между процессами через отображаемую память и семафоры. Функциональность инвертирования строк и записи в файлы реализована корректно. Были приобретены практические навыки работы с отображаемыми файлами, семафорами и обработкой ошибок. Программа протестирована на операционной системе Linux и показала стабильную работу.