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: Татаркин И. В.

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:** 1. Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. 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 системных сигналов/событий.
3. Обрабатывать системные ошибки, которые могут возникнуть в результате работы программы.
4. Реализовать суммирование строк в дочернем процессе и записывает результат в файл, заданный пользователем.

**Описание решения:**

Программное решение состоит из трех основных компонентов:

1. **Родительский процесс (Parent):**
   * Инициализирует именованные семафоры и отображаемую память.
   * Создает дочерний процесс.
   * Осуществляет чтение чисел из потока ввода и передает их в отображаемую память.
2. **Дочерний процесс (Child):**
   * Получает доступ к отображаемой памяти.
   * Считает числа, складывает их с помощью функции Calculate\_sum и записывает результат в файл, заданный пользователем.
   * Завершает выполнение при получении специального сигнала "done".
3. **Вспомогательные модули (utils.c):**
   * Содержат реализацию вспомогательных функций, таких как Calculate\_sum.

**Логика работы программы:**

1. Родительский процесс создает отображаемую память с помощью shm\_open и mmap.
2. Пользователь вводит имя файла, который передается в дочерний процесс в качестве аргументов.
3. Родительский процесс создает дочерний процесса с помощью fork и запускает с его помощью execl.
4. Родительский процесс:
   * Принимает числа от пользователя.
   * Помещает числа в отображаемую память.
   * Сигнализирует дочерним о доступности данных.
5. Дочерний процесс:
   * Читает числа из отображаемой памяти.
   * Складывает числа и записывает их в файлы.
   * Возвращают управление родительскому процессу.

**Репозиторий:** https://github.com/IvanTatarkin/OS-Labs/tree/LW1/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 чисел и запись в файлы).
4. **utils.c:** Реализация вспомогательных функций, включая Calculate\_sum.

**Пример функции Calculate\_sum:**

int calculate\_sum(const char \*input) {

int sum = 0;

int num;

char \*input\_copy = strdup(input);

if (input\_copy == NULL) {

return 0;

}

char \*token = strtok(input\_copy, " ");

while (token != NULL) {

num = atoi(token);

sum += num;

token = strtok(NULL, " ");

}

free(input\_copy);

return sum;

}

**Пример работы:**./lab3

Введите имя файла (или введите 'exit' для завершения): test.txt

Теперь введите несколько групп чисел. Введите 'done', чтобы завершить ввод и выполнить обработку.

Введите группу чисел через пробел (или 'done' для обработки): 4 5 6 -9

Введите группу чисел через пробел (или 'done' для обработки): 0 2 3

Введите группу чисел через пробел (или 'done' для обработки): done

Child process started.

Содержимое shared\_mem:

test.txt

4 5 6 -9

0 2 3

Обработка файла: test.txt

Входные данные для calculate\_sum: 4 5 6 -9

Результат calculate\_sum: 6

Входные данные для calculate\_sum: 0 2 3

Результат calculate\_sum: 5

Дочерний процесс завершен, результаты записаны в test.txt.

Родительский процесс завершён обработку для файла 'test.txt'. Вы можете ввести новые данные.

Введите имя файла (или введите 'exit' для завершения):

exit

**Вывод:** В ходе выполнения лабораторной работы были решены все заданные задачи. Программа правильно организует взаимодействие между процессами через механизмы отображаемой памяти и семафоров. Функции инвертирования строк и записи данных в файлы работают без ошибок. В процессе работы я приобрел практические навыки работы с отображаемыми файлами, синхронизацией процессов с помощью семафоров и обработкой возможных ошибок. Программа была протестирована в среде Linux и показала стабильную работу без сбоев.