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»**

Группа: М8О-209Б-23

Студент: Магомедов Э.Х

Преподаватель: Миронов Е.С.

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

Дата: 26.12.24

Москва, 2024

**Постановка задачи**

**Вариант 10.**

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.

**Общий метод и алгоритм решения**

Управление процессами:

* pid\_t fork(void) – создает дочерний процесс, который является копией родительского
* int execl(const char \*path, const char \*arg, ...) – заменяет текущий процесс новой программой
* pid\_t waitpid(pid\_t pid, int \*status, int options) – ожидает изменения состояния дочернего процесса

Операции с файлами:

* int open(const char \*pathname, int flags, mode\_t mode) – открывает или создает файл
* int close(int fd) – закрывает файловый дескриптор
* int ftruncate(int fd, off\_t length) – изменяет размер файла
* int unlink(const char \*pathname) – удаляет файл из файловой системы
* FILE\* fopen(const char \*pathname, const char \*mode) – открывает поток для работы с файлом
* int fclose(FILE \*stream) – закрывает поток

Управление памятью:

* void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset) – отображает файл в память
* int munmap(void \*addr, size\_t length) – удаляет отображение памяти
* int msync(void \*addr, size\_t length, int flags) – синхронизирует память с физическим хранилищем

Операции с семафорами:

* sem\_t \*sem\_open(const char \*name, int oflag, mode\_t mode, unsigned int value) – создает/открывает семафор
* int sem\_close(sem\_t \*sem) – закрывает семафор
* int sem\_wait(sem\_t \*sem) – блокирует семафор (ожидание)
* int sem\_post(sem\_t \*sem) – разблокирует семафор (освобождение)
* int sem\_unlink(const char \*name) – удаляет именованный семафор

Работа со временем:

* usleep(unsigned int usec) – приостанавливает выполнение на заданное количество микросекунд

Информация о файлах:

* int fstat(int fd, struct stat \*statbuf) – получает информацию о состоянии файла

В целом метод решения идейно ничем не отличается от первой лабы. Просто здесь вместо пайпов используются MMF.

1. Вводим названия файлов
2. Подготавливаем файлы для MM и делаем MM
3. Инициализируем семафоры
4. Форкаем процесс для двух детей
5. Принимаем строки и направляем их в детей

Дочерние процессы в свою очередь открывают мапед файлы, а так же файлы для записи. Ну и удаляют гласные

**Код программы**

**https://github.com/EATSLEEPSESH/os\_labs/tree/lab/os\_3**

**Вывод**

Интересно было взаимодействовать с MMF файлами, как с альтернативным подходом к решению задачи в первой лабораторной работоты.