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-215Б-23

Студент: Шаталов М.А.

Преподаватель: Миронов Е.С.

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

Дата: 31.10.24

Москва, 2024

**Постановка задачи**

**Вариант 3.**

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами. parent child1 pipe1 In/out User In Out child2 In Out File Open with write mode pipe2 File Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в дочерние процессы в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: с вероятностью 80% строки отправляются в первый процесс, иначе во второй. Дочерние процессы удаляют все гласные из строк.

Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* int execl(const char \*path, const char \*arg, ...); - загружает и исполняет новый образ программы.
* close(int fd); - закрыть файл
* open(const char \*pathname, int flags, mode\_t mode); - открытие\создание файла
* exit(int status); - завершения выполнения процесса
* int shm\_open (const char \*name, int oflag, mode\_t mode); - создать или открыть разделяемую память
* int shm\_unlink (const char \*name);- удалить разделяемую память по имени
* int ftruncate (int fd, off\_t length); - изменить размер открытого файла
* void \*mmap (void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);-сопоставить область памяти с файлом
* int munmap (void \*addr, size\_t length); - отменить сопоставление области памяти
* sem\_t \*sem\_open (const char \*name, int oflag); - создать или открыть именованный семафор
* int sem\_post (sem\_t \*sem); - сигнализировать (разблокировать) семафор
* int sem\_wait (sem\_t \*sem); - ожидать (заблокироваться) на семафоре
* int sem\_unlink (const char \*name); - удалить именованный семафор

**Алгоритм работы программы**

Разделим алгоритм программы на этап инициализации и основной этап.

В блоке инициализации мы создаем и настраиваем отображаемые в память файлы(memory mapping), вводим имена файлов, создаем дочерние процессы, семафоры и так далее. В дочернем файле нужно будет повторить все действия настройки из родительского, за исключением того, что доченим процессам не требуется заново создавать файл отображения.

Дочерним процессам необходимо знать имена своих файлов, имя именного семафора и имя отображаемого файла для получения данных от родительского процесса. Все эти данные мы передадим из родительского процесса через флаги запуска программы child: int argc, char \*argv[].

Основная идея работы программы это два бесконечных цикла. Один в родительском процессе: он принимает строки и с вероятностью перенаправляет её либо в pipe 1 либо в pipe 2. Цикл в дочернем процессе считывает строку из стандартного потока входа и выполняет действие со строкой.

Условием выхода из циклов будет строка “q”, обозначающая quit. После неё прерывается основной цикл программы, и в дочерние программы отправляется сигнал о завершении работы.

После завершения работы важно дождаться завершения работы детей и закрыть открытые дескрипторы.

После завершения работы результат программы будет записан в файлы, чьи имена были переданы в начале работы родительского процесса.

Вероятность для отправки строк в разные пайпы вычисляем с помощью функции rand().

**Код программы**

**main.c**

**Протокол работы программы**

**Здесь нужно показать тесты программы (текст или скриншоты), а затем показать полный вывод утилиты strace (или какой-либо другой утилиты на Windows, если вы выполняете лабы на этой операционной системе).**

**В strace нужно обязательно выделить, где происходят системные вызовы, которые вы использовали в лабораторной работе (например, где в первой лабораторной работе был вызван fork и другие вызовы). Полный список вызовов, которые нужно будет выделить в выводе strace, будет указан при выдаче лабы в нашем канале.**

**Тестирование:**

root@3d4dec048ee8:/workspaces/MAI\_OS\_Labs/lab2/src# ./main

Enter file name for child1: f\_1

Enter file name for child2: f\_2

Enter linens. To finish entering lines, enter '!q'.

door

apple

tomato

table

answer

elefant

poor

rich

omega

q

root@3d4dec048ee8:/workspaces/MAI\_OS\_Labs/lab2/src# cat f\_1

tmt

tbl

lfnt

mg

root@3d4dec048ee8:/workspaces/MAI\_OS\_Labs/lab2/src# cat f\_2

dr

ppl

nswr

pr

rch

**Strace:**

**Вывод**

**Вывод 3-4 предложения. Также опишите проблемы, с которыми столкнулись при выполнении лабораторной работы (если они были), пожелания и т.д.**