Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу
«Операционные системы»

Студент: Хомяков Иван Андреевич
Группа: М8О-207Б-21
Вариант: 19
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/EbumbaE/OS_LAB/lab4

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Освоение принципов работы с файловыми системами
- Обеспечение обмена данных между процессами посредством технологии «File mapping»

Задание

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.

Лабораторная работа №2:

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 каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Блок вариантов 5: 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. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Вариант 19: Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2. Дочерние процессы удаляют все гласные из строк.

Общие сведения о программе

Программа компилируется из файла main.c. Также используется заголовочный файл: main.h – описывает функции и подключает include.h, work_with_event.h – описывает функции, для работы с ивентами. Также программа собирает child1 и child2, которые используют include.h, work_with_event.h и work_with_file.h (подключает нужные для работы с файлами библиотеки и описывает несколько функций для работы с файлом по заданию). work_with_file.c реализует функции, описываемые в work_with_file.h. work_with_event.c реализует функции, описываемые в work_with_file.h.

В программе используются следующие системные вызовы:

1. CreateFileMapping – создает или открывает именованный или безымянный объект сопоставления файлов для указанного файла. Нам для задания необходимо указывать имя, размер буфера и флаг INVALID_HANDLE_VALUE для поддержки системным файлом подкачки, а не файлом в файловой системе.
2. MapViewOfFile – сопоставляет представление сопоставления файла с адресным пространством вызывающего процесса, передаем в нее наш файловый дескриптор, FILE_MAP_ALL_ACCESS – сопоставляем представление файла для чтения/записи.
3. OpenFileMapping - открывает именованный объект сопоставления файлов. Указываем имя и тип разрешения.
4. _tcscpy – копирует значение строки в переменную типа TCHAR.
5. CreateEvent – создает именованный или неименованный объект события. В нашем случае указываем имя и стандартный дескриптор безопасности.
6. OpenEvent – открывает существующий именованный объект события. Указываем имя и право доступа EVENT_ALL_ACCESS для всех прав доступа к ивенту.
7. SetEvent – устанавливает указанный объект события в сигнальное состояние. Нам эта функция нужна для синхронизации и общения ребенка и родителя.
8. WaitForSingleObject - ожидает, пока указанный объект не перейдет в сигнальное состояние или пока не истечет время ожидания. Указываем handler ивента и время ожидания INFINITE. Используем эту функцию, чтобы дождаться завершения нашего ребенка.
9. WaitForMultipleObjects - ожидает, пока один или все указанные объекты не перейдут в сигнальное состояние или пока не истечет время ожидания. Указываем количество, массив дескрипторов событий и время ожидания INFINITE. Используем эту функцию, чтобы ребенок определился, выполняться ему или сразу закрыться. Т.к. мы указали дескриптор запуска ребенка первым, достаточной проверкой будет ‘возвращаемый функцией код != WAIT_OBJECT_0’. Т.е., если нам вернется все, кроме кода запуска ребенка, мы закрываем его.
10. CreateProcess – создает процесс, принимает кучу параметров, важные из них (szCmdline – командная строка, запускающая child.exe; TRUE – наследуем pipe; siStartInfo – сами pipe;

siProcInfo – устанавливаем информацию о процессе)

11. CloseHandle – закрываем handle

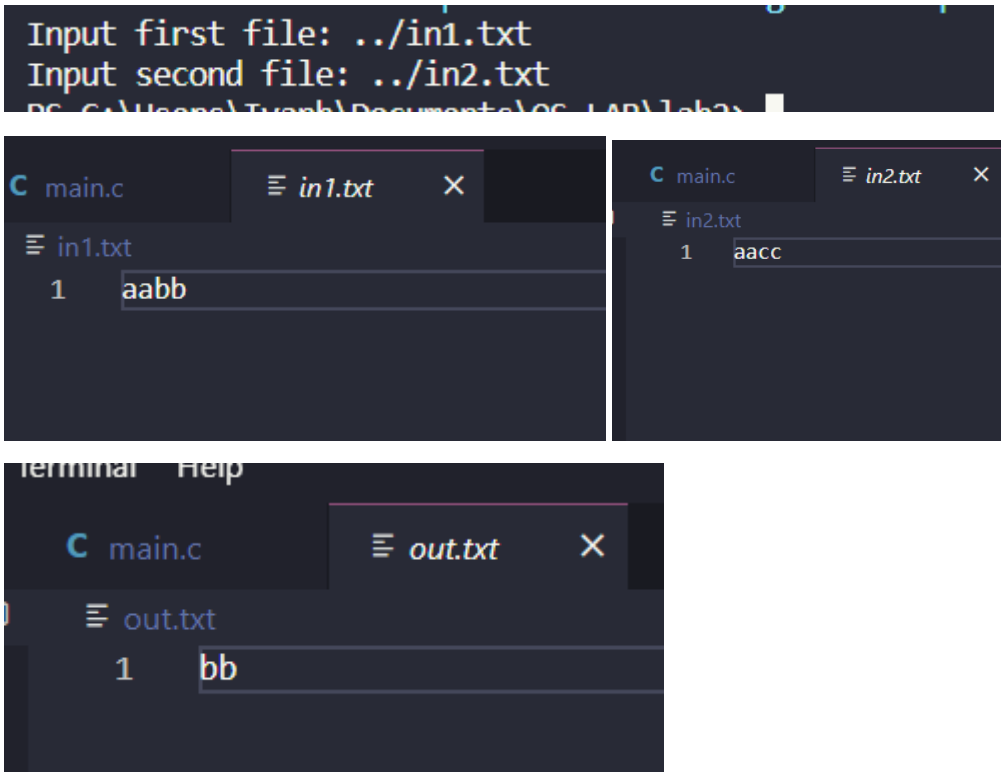
Общий метод и алгоритм решения

Родительский процесс создает два дочерних процесса child1 и child2. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано child1. Аналогично для второй строки и процесса child2. Родительский и дочерние процессы представлены разными программами. Родительский с вероятностью 80% выбирает child1, иначе child2. Он записывает имя нужного файла для ребенка в файл с отображением памяти и сигнализирует одному из них о начале работы, другому о завершении. Дочерний процесс либо открывает два файла, один читает, в другой записывает только согласные буквы, после чего сигнализирует родителю о своем завершении, либо сразу закрывается.

Исходный код

В репозитории.

Демонстрация работы программы



Выводы

Научился создавать дочерние процессы, ивенты для общения процессов и файлы с отображением памяти для передачи информации между процессами.