**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Операционные системы»**

**Тема: Межпроцессное взаимодействие**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Яловега Н.В. |
| Преподаватель |  | Тимофеев А.В. |

Санкт-Петербург

2021

**Введение**

Цель работы: Исследовать инструменты и механизмы взаимодействия процессов в Windows.

Задание 1: Реализация решения задачи о читателях-писателях.

1. Выполнить решение задачи о читателях-писателях, для чего необходимо разработать консольные приложения «Читатель» и

«Писатель»:

* одновременно запущенные экземпляры процессов-читателей и процессов-писателей должны совместно работать с буферной памятью в виде проецируемого файла:
  + размер страницы буферной памяти равен размеру физической страницы оперативной памяти;
  + число страниц буферной памяти равно сумме цифр в номере студенческого билета без учета первой цифры.
* страницы буферной памяти должны быть заблокированы в оперативной памяти (функция **VirtualLock**);
* длительность выполнения процессами операций «чтения» и

«записи» задается случайным образом в диапазоне от 0,5 до 1,5 сек.;

* для синхронизации работы процессов необходимо использовать объекты синхронизации типа «семафор» и «мьютекс»;
* процессы-читатели и процессы-писатели ведут свои журнальные файлы, в которые регистрируют переходы из одного «состояния» в другое (начало ожидания, запись или чтение, переход к освобождению) с указанием кода времени (функция **TimeGetTime**). Для состояний «запись» и «чтение» необходимо также запротоколировать номер рабочей страницы.

1. Запустите приложения читателей и писателей, суммарное количество одновременно работающих читателей и писателей должно быть не менее числа страниц буферной памяти. Проверьте функционирование приложений, проанализируйте журнальные файлы процессов, постройте сводные графики смены «состояний» для не менее

5 процессов-читателей и 5 процессов-писателей, дайте свои комментарии относительно переходов процессов из одного состояния в другое. Постройте графики занятости страниц буферной памяти (проецируемого файла) во времени, дайте свои комментарии.

1. Подготовьте итоговый отчет с развернутыми выводами по заданию.

Задание 2: Использование именованных каналов для реализации сетевого межпроцессного взаимодействия.

1. Создайте два консольных приложения с меню (каждая выполняемая функция и/или операция должна быть доступна по отдельному пункту меню), которые выполняют

* приложение-сервер создает именованный канал (функция Win32 API – **CreateNamedPipe**), выполняет установление и отключение соединения (функции Win32 API – **ConnectNamedPipe**, **DisconnectNamedPipe**), создает объект «событие» (функция Win32 API – **CreateEvent**) осуществляет ввод данных с клавиатуры и их асинхронную запись в именованный канал (функция Win32 API – **WriteFile**), выполняет ожидание завершения операции ввода- вывода (функция Win32 API – **WaitForSingleObject**);
* приложение-клиент подключается к именованному каналу (функция Win32 API – **CreateFile**), в асинхронном режиме считывает содержимое из именованного канала файла (функция Win32 API – **ReadFileEx**) и отображает на экран.

1. Запустите приложения и проверьте обмен данных между процессами. Запротоколируйте результаты в отчет. Дайте свои комментарии в отчете относительно выполнения функций Win32 API
2. Подготовьте итоговый отчет с развернутыми выводами по заданию.

**Задание 1. Реализация решения задачи о читателях-писателях.**

Для выполнения задания воспользуемся тремя программами: программа-читатель, программа-писатель и программа, запускающая первые две.

Программа для запуска создает все нужные файлы, проекцию файла для работы процессов и запускает сами процессы, затем ожидает остановки всех процессов и освобождает ресурсы.

**Мьютекс** (англ. mutex, от **mut**ual **ex**clusion — «взаимное исключение») — это базовый механизм синхронизации. Он предназначен для организации взаимоисключающего доступа к общим данным для нескольких потоков с использованием барьеров памяти (для простоты можно считать мьютекс дверью, ведущей к общим данным).

**Захват мьютекса:** Поток запрашивает *монопольное использование* общих данных, защищаемых мьютексом. Дальше два варианта развития событий: происходит захват мьютекса этим потоком (и в этом случае ни один другой поток не сможет получить доступ к этим данным) или поток блокируется (если мьютекс уже захвачен другим потоком).

**Освобождение мьютекса:** Когда ресурс больше не нужен, текущий владелец должен вызвать функцию разблокирования, чтобы и другие потоки могли получить доступ к этому ресурсу. Когда мьютекс освобождается, доступ предоставляется одному из ожидающих потоков.

**Семафор** – это объект, который используется для контроля доступа нескольких потоков до общего ресурса. В общем случае это какая-то переменная, состояние которой изменяется каждым из потоков. Текущее состояние переменной определяет доступ к ресурсам.

**Выводы по заданию**

В данном задании были использованы такие объекты синхронизации, как семафоры и мьютексы. В программе создаётся два семафора – для «чистых страниц», то есть не обработанных писателем, и «использованных», записанных писателем и ещё не прочитанных читателем.

В начале работы программы значение семафора для «чистых» страниц становится равно N, в то время как для «использованных» - 0. Таким образом писатели могут сразу приступать к работе – читатели ожидают появления «использованных» страниц.

Писатель ожидает сигнала от семафора «чистых страниц» - так он понимает, что есть страница, которую он может записать, а также мьютекс. Затем писатель выполняет необходимые действия. Наконец, выполняется ReleaseSemaphore. Работа писателя завершается.

Для читателей ситуация аналогичная, только ожидают они «использованные» страницы.

**Задание 2. Использование именованных каналов для реализации сетевого межпроцессного взаимодействия.**

Для использования именованных каналов для реализации  
сетевого межпроцессного взаимодействия было создано две программы – сервер и клиент.

Сначала необходимо запустить программу-сервер и создать именованный канал. Далее программа ожидает подключение клиента. После подключения сервер может передавать сообщения клиенту. Для этого предусмотрен отдельный пункт меню.

Для установления соединения между программами используется именованный канал (named pipe).

Для создания именованного канала Pipe можно использовать функцию CreateNamedPipe. Канал может использоваться как для записи в него данных, так и для чтения.

После того как серверный процесс создал канал, он может перейти в режим соединения с клиентским процессом. Соединение со стороны сервера выполняется с помощью функции ConnectNamedPipe.

**Выводы по заданию**

Для реализации сетевого межпроцессного взаимодействия был использован такой объект, как именованный канал.

Программа-сервер создаёт именованный канал с заданным названием и ожидает подключения клиента.

Затем запускается клиент и находит по названию нужный канал – соединение установлено.

По каналу сервер передаёт сообщение клиенту, которое выводится на экран. В любой момент можно отключить соединение и завершить работу программ.

НАПИСАТЬ ПРО КАНАЛЫ ПОДРОБНЕЕ

**Вывод**

Были исследованы инструменты и механизмы взаимодействия процессов в Windows.

**Приложения**