МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

(наименование института)

Кафедра автоматизированных систем и технологий

(наименование кафедры)

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

(наименование направления подготовки/специальности)

Дисциплина Архитектура ЭВМ

(наименование дисциплины)

Лабораторная работа №6.

Многопоточные приложения.

Выполнили студенты Копосов Лев Владимирович

Копосов Владимир Владимирович

(фамилия, имя, отчество)

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома

Вопросы:

1. Какие примитивы синхронизации вы знаете?

Изученные примитивы синхронизации:

1. мьютекс
2. семафор
3. барьер
4. критическая секция
5. событие
6. таймер
7. процесс, поток
8. Что общего у мьютекса и критической секции? Чем они отличаются?

Сходства: мьютекс и критическая секция используются для синхронизации доступа к общим ресурсам в многопоточной среде.

Отличия: мьютекс является более гибким, чем критическая секция. Он может быть использован для синхронизации между разными процессами (можно задать именованный мьютекс и использовать его из нескольких процессов), а критическая секция работает только в пределах одного процесса. Критическая секция обеспечивает автоматическую блокировку и разблокировку при входе и выходе из участка кода, где нужен доступ к ресурсам. Мьютекс требует явной блокировки и разблокировки

В чем разница между событием с автосбросом и ручным сбросом?

Событие с автосбросом автоматически сбрасывается в неактивное состояние после того, как один поток, ожидающий его события, был разблокирован. Другие потоки не будут разблокированы, пока событие снова не будет активировано.

Событие с ручным сбросом остается активным после того, как один поток, ожидающий его события, был разблокирован. То есть можно разблокировать несколько потоков, прежде чем событие будет вручную сброшено в неактивное состояние.

1. Как дождаться перехода в сигнальное состояние одного примитива? Одного из нескольких указанных примитивов? Всех указанных примитивов?

Для ожидания перехода в сигнальное состояние одного примитива можно использовать функцию WaitForSingleObject() или механизмы синхронизации. Например: мьютекс и семафор.

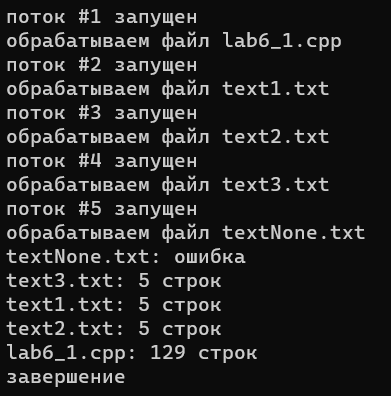
Для ожидания перехода в сигнальное состояние одного из нескольких указанных примитивов, нужно использовать функцию WaitForMultipleObjects() с параметром bWaitAll = FALSE или механизмы синхронизации. Например: семафор.

Для ожидания перехода в сигнальное состояние всех указанных примитивов, нужно использовать функцию WaitForMultipleObjects() с параметром bWaitAll = TRUE или механизмы синхронизации. Например: семафор.

Вывод в консоль.

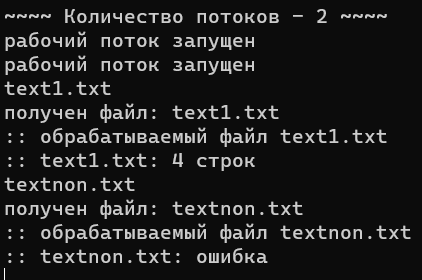
Задание 1.

Создайте программу, выполняющую подсчёт числа строк в текстовых файлах с помощью синхронного ввода/вывода и вывод этого числа в консоль. Операция должна выполняться в отдельных потоках, по одному на файл, в то время как основной поток должен дожидаться его завершения.  
Выделите отдельную функцию подсчёта, принимающую на вход имя файла, и возвращающую число строк в нём. Также выделите отдельную функцию, которая будет телом потока. При выводе в консоль синхронизируйте операции вывода с помощью мьютекса.



Задание 2.

Модифицируйте программу следующим образом: главный поток должен читать имена файлов для обработки из стандартного потока ввода (cin). Не следует считать, что вы сможете прочитать все имена файлов заранее - они могут подаваться на вход как сразу, так и постепенно. Очередное имя должно помещаться в переменную, откуда оно будут скопировано одним из нескольких (количество должно задаваться константой) рабочих потоков. Вывод результатов обработки (число строк или текст ошибки) должен производиться в стандартный вывод (cout). Чтобы избежать перекрытия вывода от разных потоков, используйте мьютекс.



Дополнительное задание.

Модифицируйте программу из задания 2 таким образом, чтобы она принимала число рабочих потоков как необязательный аргумент командной строки. Если аргумент не задан, программа должна создавать 4 рабочих потока.

