Лабораторная работа 07

Синхронизация

OC, ПОИТ-3

**Задание 01. Windows**

1. Напишите ассемблерный код с применением команд BTS или BTR, демонстрирующий реализацию механизма синхронизации двух потоков одного процесса и поясните его работу.

**Задание 02. Windows**

1. Разработайте приложение **OS07\_02**, запускающее два дочерних потока **A** и **B**.
2. Все потоки выполняют циклы в 90 итераций, выводящие имена потоков и номера итерации с задержкой в 0.1 сек.
3. Приложение **OS07\_02** синхронизирует выполнение потоков **main**, **A** и **B** с помощью механизма **critical section.**
4. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 в каждом потоке.

**Задание 03. Windows**

1. Разработайте приложение **OS07\_03**, запускающее два дочерних процесса **OS07\_03A** и **OS07\_03B** свыводом в отдельные консоли.
2. Все процессы выполняют циклы в 90 итераций, выводящие имена процессов и номера итерации с задержкой в 0.1 сек.
3. Приложение **OS07\_03** синхронизирует выполнение процессов **OS07\_03**, **OS07\_03A** и **OS07\_03B** с помощью механизма **mutex.**
4. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60.

**Задание 04. Windows**

1. Разработайте приложение **OS07\_04**, запускающее два дочерних процесса **OS07\_04A** и **OS07\_04B** свыводом в отдельные консоли.
2. Все процессы выполняют циклы в 90 итераций, выводящие имена процессов и номера итерации с задержкой в 0.1 сек.
3. Приложение **OS07\_04** синхронизирует выполнение процессов **OS07\_04**, **OS07\_04A** и **OS07\_04B** с помощью механизма **semaphore.**
4. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 одного (любого) процесса и двух других процессов. Другими словами, итерации с 30 по 60 должны одновременно выполняться только в двух из трех процессов.

**Задание 05. Windows**

1. Разработайте приложение **OS07\_05**, запускающее два дочерних процесса **OS07\_05A** и **OS07\_05B** свыводом в отдельные консоли.
2. Все процессы выполняют циклы в 90 итераций, выводящие имена процессов и номера итерации с задержкой в 0.1 сек.
3. Приложение **OS07\_05** синхронизирует выполнение процессов **OS07\_05**, **OS07\_05A** и **OS07\_05B** с помощью механизма **event**.
4. Синхронизация должна обеспечивать выполнение приложения в два этапа:
5. выполнение итераций с 1 по 15 процесса **OS07\_05**;
6. одновременное выполнение всех трех процессов: **OS07\_05** – продолжает выполнение итераций; процессы **OS07\_05A** и **OS07\_05B** выполняются начиная с первой итерации.

**Задание 06.Linux**

1. Разработайте приложение **OS07\_06**, запускающее два дочерних потока **A** и **B**.
2. Все потоки выполняют циклы в 90 итераций, выводящие имена потоков и номера итерации с задержкой в 0.1 сек.
3. Приложение **OS07\_06** синхронизирует выполнение потоков **main**, **A** и **B** с помощью механизма **mutex.**
4. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 в каждом потоке.

**Задание 07.** Ответьте на следующие вопросы

1. Дайте определение понятию «синхронизация потоков».
2. Объясните понятие «взаимная блокировка».
3. Перечислите механизмы авторизации OS.
4. Поясните в чем разница между механизмом **mutex** и **semaphore**.
5. Почему **mutex,** **semaphore, event** создают объект ядра OS, а **critical section** нет.

1. Дайте определение понятию «синхронизация потоков».

синхронизация - механизм упорядочивания выполнения программных блоков двух или более потоков.

Синхронизация потоков – механизм, позволяющий потокам согласовывать свою работу с общими ресурсами. Этот механизм представляет собой набор объектов операционной системы, которые создаются и управляются программно, являются общими для всех нитей в системе и используются для координирования доступа к ресурсам.

2. Объясните понятие «взаимная блокировка».

Взаимная блокировка (deadlock) - явление при котором все потоки находятся в режиме ожидания. Происходит, когда достигаются состояния:

● взаимного исключения: по крайней мере один ресурс занят в режиме неделимости и, следовательно, только один поток может использовать ресурс в любой данный момент времени.

● удержания и ожидания: поток удерживает как минимум один ресурс и запрашивает дополнительные ресурсов, которые удерживаются другими потоками.

● отсутствия предочистки: операционная система не переназначает ресурсы: если они уже заняты, они должны отдаваться удерживающим потокам сразу же.

● цикличного ожидания: поток ждет освобождения ресурса другим потоком, который в свою очередь ждет освобождения ресурса заблокированного первым потоком.

3. Перечислите механизмы авторизации OS.

- Critical section ;

- Mutex;

- Semaphore;

- Atomic operation (interlocking function)

- Event;

- Writable timer.

4. Поясните в чем разница между механизмом mutex и semaphore.

В основе семафора лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, мьютекс можно представить в виде переменной, которая может находиться в двух состояниях: в заблокированном и в незаблокированном. При входе в свою критическую секцию поток вызывает функцию перевода мьютекса в заблокированное состояние. При выходе из критической секции поток вызывает функцию перевода мьютекса в незаблокированное состояние.

5. Почему mutex, semaphore, event создают объект ядра OS, а critical section нет.

Critical section – механизм синхронизации нескольких потоков одного процесса, НЕ ЯВЛЯЕТСЯ объектом ядра OS.