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».

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

1. Объясните понятие «взаимная блокировка».

**Ситуация при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжать свое выполнение.**

1. Перечислите механизмы авторизации OS.

**mutex,** **semaphore, event, critical section**

1. Поясните в чем разница между механизмом **mutex** и **semaphore**.

**Разница только в том, что мьютекс объекта может захватить одновременно только один поток, а в случае с семафором используется счетчик потоков, и доступ к ресурсу могут получить сразу несколько из них.**

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

**mutex,** **semaphore, event могут быть использованы несколькими процессами одновременно, критическая секция же принадлежит процессу и служит для синхронизации только его потоков.**