Лабораторная работа № 5

“ Алгоритмы синхронизации процессов.”

Цель

Изучить алгоритмы синхронизации процессов и средства операционной системы, позволяющие осуществлять синхронизацию.

Общие задачи

1. *Написать диспетчер процессов, который бы постоянно держал в памяти указанное количество экземпляров заданного (командной строкой) приложения. В случае, если экземпляр приложения работает более заданного времени, то диспетчер должен принудительно завершить данный экземпляр приложения. Диспетчер должен выставлять себе приоритет выше, чем приоритет запускаемых приложений. Вывод приложений должен игнорироваться. При завершении основной программы завершить дочерние процессы*
2. *Написать программу p, которая бы при запуске создавала два дочерних процесса p1 и p2. Программа p должна генерировать случайные числа и записывать их в буфер. Процессы p1 и p2 должны читать числа из буфера, p1 должен выводить числа на консоль, а p2 — в файл. Взаимодействие между процессами реализовать с помощью объектов ядра.*
3. *Написать программу, копирующую файл. Программа должна запускать два дочерних процесса, один читает файл, другой пишет. Передача данных между процессами должна быть реализована через общую область памяти, синхронизация – с помощью объектов ядра. Материнский процесс должен обрабатывать ошибки в дочерних процессах (например, если при записи произошла ошибка – читающий процесс должен завершаться материнским).*
4. *Требуется написать программу для скачивания изображений. После запуска программа ждет ввода пользователя в виде пути к изображению, после чего пытается его скачать. Процесс скачивания выполняется таким образом, что, текущих закачек может быть несколько (каждая должна запускаться в отдельном потоке). Каждая операция, выполняемая процессом закачки, должна записываться в специальный лог файл.*
5. *Создать консольное приложение, порождающее несколько потоков. Первый поток высчитывает таблицу значений функции (можно взять любую математическую функцию), второй поток осуществляет вывод каждого значения в файл по мере его высчитывания. Третий поток ведет лог процедуры обмена, записывая туда время получения значения функции и время записи данного значения в файл. Каждая пара значений, полученная в процессе вычисления, должна быть занесена в объект класса Point, который может быть уничтожен только тогда, когда информация о нем будет занесена в лог-файл. Обращение к объекту Point должно происходить через потокобезопасный умный указатель.*
6. *Обедающие философы (вариация). Создается пять процессов (по одному на философа). Процессы разделяют пять переменных (вилок). Каждый*

процесс находится только в двух состояниях - либо он "размышляет", либо "ест спагетти". Чтобы начать "есть", процесс должен взять "две вилки" (захватить две переменные). Закончив "еду", процесс освобождает захваченные переменные и начинает "размышлять" до тех пор, пока снова "проголодается".

7. *У парикмахера есть одно рабочее место и приемная с несколькими стульями. Когда парикмахер заканчивает подстригать клиента, он отпускает клиента и затем идет в приёмную, чтобы посмотреть, есть ли там ожидающие клиенты. Если они есть, он приглашает одного из них и стрижет его. Если ждущих клиентов нет, он возвращается к своему креслу и спит в нем. Каждый приходящий клиент смотрит на то, что делает парикмахер. Если парикмахер спит, то клиент будит его и садится в кресло. Если парикмахер работает, то клиент идет в приёмную. Если в приёмной есть свободный стул, клиент садится и ждёт своей очереди. Если свободного стула нет, то клиент уходит.*
   Решение должно гарантировать, что парикмахерская функционирует правильно с парикмахером, стригущим любого пришедшего, пока есть клиенты, и затем спящим до появления следующего клиента.

Варианты объектов синхронизации

1. Для выполнения задания используем критические секции.
2. Для выполнения задания используем семафоры.
3. Для выполнения задания используем мьютексы.
4. Для выполнения задания используем события.
5. Для выполнения задания используем ожидаемые таймеры.
6. Для выполнения задания реализовать безопасную очередь на основе одного из предложенных в вариантах заданий объектов синхронизации.

Вы должны выбрать наиболее подходящий вариант объекта синхронизации одного из предложенных вам заданий. Эффективность выбора будет учитываться при защите лабораторной работы.

Технические требования

Реализация может быть выполнена под управлением операционной системы Linux или Windows. Для синхронизации должны быть использованы объекты уровня операционной системы. Не допускается использование нестандартных библиотек C++ или сторонних фреймворков.

Что должно быть в отчете

1. Краткое описание задачи.
2. Описание методов, которые был использованы для решения задачи.
3. Обосновать необходимость использования конкретного объекта синхронизации для решения вашей задачи.
4. Схемы алгоритмов решения поставленной задачи.

Этапы защиты лабораторной

1. Показать отчет и оформленные исходные тексты лабораторной работы.
2. Продемонстрировать работу программы.
3. Ответить на контрольные вопросы.

Контрольные вопросы

1. Поясните, в чем состоит проблема синхронизации процессов/потоков?
2. Что такое “эффект гонок”?
3. Что такое клинч?
4. Что такое взаимоисключение?
5. Что такое критическая секция с точки зрения теории операционных систем?
6. Что такое объект ядра? Какие объекты ядра операционной системы windows или Linux вы знаете?
7. Какие есть подходы к предотвращению, последствий тупиков?
8. Как можно диагностировать наличие в системе большой очереди процессов или тупика? В чем различие этих ситуаций?
9. Чем объект синхронизации критическая секция отличается от объекта синхронизации семафор в операционной системе windows.
10. Что такое файлы, проецируемые в память? Какие базовые механизмы операционных систем реализованы через данный подход?

Рекомендуемая литература:

1) Рихтер Джеффри. Windows для профессионалов. Создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows/Пер. с англ. — 4-е изд. — СПб: Питер, 2008. — 743 с.: ил. ISBN 5-272-00384-5.
2) Системное программирование в Windows 2000 для профессионалов — СПб: Питер, 2001. — 624 с.: ил. ISBN 5-272-00152-4.
3) Операционная система UNIX®. — СПб: БХВ — Санкт-Петербург, 2000. — 528 с.: ил. ISBN 5-8206-0030-4.
4) Ресурсы man ОС Linux/UNIX. Точки входа: apropos pthread; apropos shm; man fork; man exec; apropos uid; apropos euid.