Системное программирование
Л.р.3. Управление потоками и нитями

Цель:

Концепция вычислительных потоков и ее реализация в Windows (также «нити» и кооперативная многозадачность). Основные этапы жизненного цикла потоков и элементарное управление ими: порождение, завершение, получение и изменение состояния. Типичные (простые) ситуации многопоточности, типичное (простое) использование многопоточности.

Теоретическая и методическая часть

Вычислительные потоки. Вычислительный поток как системный объект, объект Thread в Windows. Атрибуты потока, в т.ч. приоритеты. Состояния потока. Разделяемые адресное пространство и пространство дескрипторов.

Создание (порождение) потоков, главный поток. Диспетчирование потоков. Приостановка и возобновление потоков. Завершение потоков. Код завершения.

Нити (Fiber), особенности их выполнения. Взаимосвязь нитей и потоков. Создание, завершение, переключение нитей.

Программный интерфейс (API) для «базового» управления потоками и нитями.

Практическая часть

Общая постановка задачи:

Приложение, демонстрирующее простые ситуации, связанные с управлением потоками, в основном их создание, завершение, получение информации о текущем состоянии, манипулирование приоритетами, а также с оценкой влияния на производительность приложения и на работу системы в целом. (Предполагается, что решение задачи не требует обязательно использовать специализированные механизмы взаимодействия.)

Специальных требований к приложениям не предъявляется; в частности, во многих случаях они могут быть не обязательно оконными, но также и консольными.

Варианты заданий:

– Приоритеты
– Многопоточная сортировка
– Многопоточная работа с файлом
– Сравнение многопоточной и многонитевой реализаций

– Нагрузочная способность («стресс-тест»)
– ...

1 Приоритеты

Запуск нескольких (можно фиксированное количество) потоков с разными приоритетами (можно задать заранее) и оценка их производительности. Желательно отображение времени работы потока – в течение его выполнения и итоговое после завершения, и объема выполненной потоком работы – в простейшем случае счетчик итераций. «Содержимое» потока – произвольная задача с достаточной вычислительной сложностью (сортировка, умножение матриц и т.п.)

Дополнение: анализ поведения при наличии высокоприоритетных потоков.

2 Многопоточная сортировка

Разбиение массива на несколько частей (фрагментов), сортировка каждого отдельным потоком, окончательная «сборка». Количество потоков (в т.ч. единственный поток), размер массива задаются пользователем, количество потоков не слишком большое, чтобы оставалось удобным для отображения.

Отображение прогресса выполнения (достаточно готовности/неготовности каждого фрагмента), время выполнения для сравнения и анализа зависимости.
Соотнесение с уровнем загруженности системы (использование ЦП).

3 Многопоточная работа с файлом

Чтение файла несколькими потоками, «сборка» результата. Количество потоков, файл для чтения – выбор пользователем. Количество потоков – аналогично предыдущему (в т.ч. единственный поток).

Оценка времени, зависимость от начальных параметров.

В дальнейшем возможно сравнение с асинхронной реализацией.

4 Сравнение многопоточной и многонитевой реализаций

Сравнение эффективности (производительности) реализации распараллеливаемого алгоритма потоками и нитями (далее в качестве примера умножение матриц, можно подобрать другие алгоритмы).

Пусть размерность матриц – $M \times M$, количество рабочих потоков (нитей) N – в 8..16 раз меньше. Каждый поток (каждая нить) выбирает свои строки и столбцы по определенному правилу (например, «блоками» подряд или чередованием через N).
После обработки очередной пары «строка-столбец» поток выполняет Sleep(1) для переключения на другой поток, нить явно передает управление другой нити.
Результат готов после завершения работы всех потоков (нитей).

Сравнение результатов, в т.ч. и с «линейной» (без распараллеливания) реализацией, соотнесение с уровнем загруженности системы (использование ЦП).

5 Нагрузочная способность («стресс-тест»)

Оценка влияния количества параллельно выполняющихся потоков на загруженности системы и ее состояние в целом, оценка предельного количества активных потоков.
Аналогично – для принудительно приостановленных (suspended) потоков.

Алгоритм функций потоков выбирается таким, чтобы минимизировать зависимости между потоками и избежать взаимных зависимостей и блокировок.

Также необходимо обеспечить достаточно плавное нарастание нагрузки, позволяющее наблюдать эффект.

6 ...