Операционные системы.
Лабораторная работа №3.

Исследование особенностей реализации многозадачности в ОС семейства Windows

Цель работы: изучение на практических примерах, как изменяется состояния потока; исследование особенностей реализации вытесняющей многозадачности в Windows; исследование влияния приоритета выполнения на работу потоков.

Методические указания.

При выполнении процесса, потоки, принадлежащие процессу, постоянно меняют своё состояние. Состояние потока меняется в соответствии с внутренней логикой работы программы, а также в связи с внешними по отношению к процессу событиями. Изменение состояния потока осуществляет операционная система. Вся необходимая для ОС информация о потоке хранится в блоке управления потоком. Опираясь на эту информацию ОС управляет потоками. Блок управления потоком ещё называется системным контекстом потока.

В основе многозадачности Windows лежит квантование времени и вытеснение. Каждому потоку выделяется на выполнение определённый промежуток времени, который называется квантом времени. После истечения кванта времени исполняющийся поток прерывается и переходит в состояние готовность. Вместо него запускается следующий поток, стоящий в очереди. Такая процедура называется вытеснением.

Возникает вопрос, каким образом останавливается выполнение потока по истечении кванта времени? Следует отметить, что операционная система сама не может прервать выполнение потока. ОС состоит из набора потоков. Чтобы поток выполнил какое-либо действие, он должен быть запущен на процессоре. Это действие будет возможно, только если выполняющийся поток будет остановлен. Таким образом, исполняющийся поток может остановить не другой поток, некая другая сила. Такой силой является прерывание.

Прерывания делятся на внутренние и внешние. Внутренние прерывания организуются внутри процессора, а внешние снаружи. Внутренние прерывания бывают двух типов: программные и исключительные ситуации. Программное прерывание создаёт исполняющаяся программа в следующих случаях: запрос к ОС для организации операции ввода/вывода, обращение к системным функциям ОС, завершение работы. Исключительные ситуации или исключения возникают в том случае, когда процессор не может выполнить некоторую инструкцию исполняющегося потока. Это происходит в следующих случаях: деление на ноль, обращение к чужому адресному пространству, недействительный код инструкции и т. д.

Внешние прерывания тоже бывают двух типов: прерывания ввода/вывода и аппаратное прерывание. Прерывание ввода вывода возникает, когда периферийное устройство сигнализирует о готовности выполнить операцию ввода/вывода. Аппаратное прерывание создаёт какое-нибудь устройство, входящее в компьютер. Это прерывание не связано с вводом/выводом.

Квант времени считает таймер реального времени. Он расположен на системной плате. По истечении кванта времени, выделенного на выполнение потока, таймер создаёт сигнал прерывания. Этот сигнал поступает на специальный вход процессора. Возникает аппаратное прерывание, процессор останавливает выполнение потока. Вместе с сигналом прерывания на процессор поступает номер прерывания. Номер прерывание определяет, что произошло, почему надо остановить выполнение текущего потока. По номеру прерывания определяется программа обработки прерывания, которая запускается вместо остановленного потока. Адреса программ обработки прерываний находятся в определённой

области памяти. По номеру прерывания процессор определяет адрес программы обработки прерывания и запускает её на выполнение. При этом происходит смена контекста.

Смена контекста заключается в записи новой информации о потоке в блок управления потоком, а также в сохранении содержимого регистров процессора в оперативную память. При возобновлении выполнения программы содержимое регистров процессора восстанавливается. В результате выполнение потока возобновляется с того же самого места, на каком было прервано.

Для исследования изменений состояний потока необходимо использовать следующие программы **procexp.exe, badapp32, badapp16.exe, counter.exe.**

При выполнении лабораторной работы требуется записывать полученные результаты. В результате выполнения лабораторной работы должен быть оформлен отчёт, который должен быть показан преподавателю. В отчёте должны содержаться объяснения полученных результатов и подробные ответы на поставленные вопросы. Для облегчения оценки выполненной работы рекомендуется по пунктам приводить текст задания и тут же результат выполнения.

Указания по выполнению лабораторной работы.

1. Исследование сбоев и зависаний программ при работе компьютера.

1.1. Для интерпретации результатов, получающихся при выполнении лабораторной работы, необходимо знать некоторые определения. Запишите в отчёт определение следующих терминов: **контекст потока**, **переключение контекста (Context Switch)**. Укажите о чём свидетельствует увеличение числа переключений контекста, а о чём наоборот отсутствие переключений контекста. Также запишите, что значит **сбой** выполнения программы. Приведите конкретные примеры ситуаций, приводящих к сбою. Напишите, что значит **зависание** программы и когда оно возникает.

1.2. Для наблюдения за процессами и потоками запустите программу Process Explorer (**procexp.exe**).

1.2. Запустите программу **badapp32.exe**. Эта программа предназначена для моделирования выполнения программой некорректной операции (сбой) и моделирования зависания программы.

1.3. Запишите приоритет процесса **badapp32** и приоритет основного потока, принадлежащего этому процессу. Для этого найдите в списке процессов процесс badapp32, щёлкните по нему правой кнопкой мыши и выберите пункт меню **Properties**. В открывшемся окне выберите закладку **Threads**.

1.4. Проследите, как меняется количество изменений контекста потока (Context Switches) если программа не активна. Каков её статус? Как меняется Context Switches при наведении курсора мыши на окно программы **badapp32** и выборе какого-либо меню? В отчёте приведите объяснение наблюдаемым явлениям.

1.5. Промоделируйте сбой программы **badapp32**, выбрав пункт меню **Action – GP-Fault**. В программе Process Explorer отследите изменение состояния потока. Каков его статус? Как меняется Context Switches? Объясните произошедшие изменения.

1.6. Объясните, почему сбой одной программы не влияет на другие программы?

1.7. Закройте программу **badapp32.exe** и запустите её вновь. Промоделируйте зависание программы, выбрав пункт меню **Action – Hang**. Запишите какой приоритет стал у процесса и потока. Запишите также, какой стал статус у потока и как меняется Context Switches? Объясните получившиеся результаты.

1.8. Объясните, почему зависание одной программы не влияет на другие программы?

2. Исследование свойств кооперативной (невытесняющей) и вытесняющей многозадачности Win32.

2.1. Попробуйте запустить программу **badapp16.exe** в 64-х разрядной операционной системе. Почему не удалось её запустить?

2.2. Запустите 3 экземпляра приложения **badapp16** в 32-х разрядной операционной системе и произведите операцию искусственного зависания (Hang) одного из них. Зафиксируйте поведение остальных экземпляров программы. Сделайте вывод и запишите его в отчёт.

2.3. Запустите 3 экземпляра приложения **badapp32** и произведите операцию искусственного зависания (Hang) одного из них. Зафиксируйте поведение остальных экземпляров программы. Объясните, получившиеся различия в поведении 16-разрядных и 32-разрядных приложений.

3. Исследование поведения процесса в режиме реального времени.
   3.1. Переведите программу **badapp32** в режим зависания (Hang).
   3.2. Используя **Диспетчер задач**, задайте процессу badapp32.exe приоритет реального времени.
   3.3. Используя программу Process Explorer, убедитесь, что приоритет потока управления равен 24.
   3.4. Запишите, какой статус у потока? Как меняется Context Switches? На каком ядре запущен поток, и какой процент процессорного времени ядра занимает поток? Объясните получившиеся результаты.

4. Определение влияния приоритета процесса на выделяемое этому процессу процессорное время.Для дальнейшей работы надо использовать программу counter. Для того, чтобы программа заработала, надо зарегистрировать в операционной системе библиотеку VB40032.DLL. Для этого надо обладать правами администратора. Если у Вас нет таких прав, то вместо counter можно использовать программу badapp32. В этом случае надо перейти к п. 5. В Windows 7 при запуске программы counter пользователем с правами администратора библиотека зарегистрируется автоматически. В Windows 10 для регистрации библиотеки надо нажать правой кнопкой мыши на файл counter.exe и в контекстном меню выбрать пункт «Запуск от имени администратора». После этого библиотека зарегистрируется, и программа counter запустится. Если программа counter запустилась, то п. 5 выполнять не надо. Запустите два экземпляра приложения **counter.** Определите при помощи утилиты «Диспетчер задач» доли процессорного времени, выделяемого каждому из полученных процессов. Процент процессорного времени, отведённому каждому процессу, представлен в колонке ЦП «Диспетчера задач».
   4.2. Измените приоритет одного процесса **counter** на **Высокий**, а другого на **Низкий** и определите доли процессорного времени, выделяемого каждому из полученных процессов. Объясните получившийся результат.
   4.3. Запустите оба процесса **counter** на одном ядре процессора. Для этого в «Диспетчере задач» надо нажать правой кнопкой мыши на процесс **counter** и выбрать из контекстного меню «**Задать соответствие»** (**Задать сходство**). Как изменился процент процессорного времени, отводимый каждому процессу counter? Почему?
   4.4. Запустите третий экземпляр программы **counter** на том же процессорном ядре, что и остальные процессы **counter**. Как изменилась доля процессорного времени, отводимого каждому процессу counter? Почему?
   4.5. Установите у нового процесса приоритет **Высокий.** Как изменилась доля процессорного времени, отводимого каждому процессу **counter**? Почему?
   4.6. Оцените качественно возможность реакции процессов **counter** на интерактивное событие – перетаскивание окна. Есть ли разница при перетаскивании разных окон **counter**? Объясните результат. Объясните, почему при перетаскивании окна **counter** останавливается счётчик.

5. Определение влияния приоритета процесса на выделяемое этому процессу процессорное время с использованием программы badapp32.
   5.1. Запустите два экземпляра приложения badapp32 и задайте режим Hang на каждом экземпляре программы. Определите при помощи утилиты «Диспетчер задач» доли процессорного времени, выделяемого каждому из полученных процессов. Процент процессорного времени, отведённому каждому процессу, представлен в колонке ЦП «Диспетчера задач».
   5.2. Измените приоритет одного процесса badapp32 на Высокий, а другого на Низкий и определите доли процессорного времени, выделяемого каждому из полученных процессов. Объясните получившийся результат.
   5.3. Запустите оба процесса badapp32 на одном ядре №3. Для этого в «Диспетчере задач» надо нажать правой кнопкой мыши на процесс badapp32 и выбрать из контекстного меню «Задать соответствие» (Задать сходство). Как изменился процент процессорного времени, отводимый каждому процессу counter? Почему?
   5.4. Запустите третий экземпляр программы badapp32 в режиме Hang на процессорном ядре №3. Как изменилась доля процессорного времени, отводимого каждому процессу counter? Почему?
   5.5. Установите у нового процесса приоритет Высокий. Как изменилась доля процессорного времени, отводимого каждому процессу badapp32? Почему?
   5.6. Оцените качественно возможность реакции процессов badapp32 на интерактивное событие – перетаскивание окна. Есть ли разница при перетаскивании разных окон badapp32? Объясните результат.