Понятие многозадачности. Процессы и потоки, приоритеты потоков. Программирование потоков.

Понятие многозадачности.

Операционные системы семейства Windows, начиная с Windows 95, поддерживают режим т.н. вытесняющей многозадачности, позволяющую нескольким приложениям работать одновременно. Время работы процессора разбивается на кванты, по истечении которых операционная система переключается его с одной задачи на другую (одна задача вытесняет другую). Т.к. кванты времени достаточно малы (около 19 мс), у пользователя создается впечатление одновременной работы нескольких приложений.

Если некоторое приложение получит квантов в два раза больше другого, то и выполняться оно будет также в два раза быстрее (условно говоря). Число выделяемых квантов определяется приоритетом задачи, который задается программистом.

Преимущества многозадачности:

Уменьшаются простои компьютера, связанные с тем, что процессор занимается решением задач, требующих продолжительного времени.

Задачи, требующие меньшего времени на выполнение, завершаются быстрее задач, чем более трудоемкие (во время решения солидной задачи можно параллельно решить несколько простеньких).

Зацикливание одной задачи не приостанавливает выполнение остальных.

Процессы и потоки.

Следует различать понятия потока и процесса.

Процесс (process) – это выполняемая программа, имеющая собственное виртуальное адресное пространство, код, данные, а также потребляемые ресурсы операционной системы (такие как файлы, окна и т.п.). Процесс порождается в момент запуска очередного приложения.

Поток (thread - нить) – это подпрограмма, выполняемая параллельно с главной программой (которая в свою очередь также является потоком – главным потоком процесса).Все потоки одного процесса имеют общее с процессом адресное пространство, обращаются к одним и тем же глобальным переменным и ресурсам процессора. Время процессора (кванты) выделяется именно потоку.

Под задачей понимается поток. Любой процесс выполняет как минимум один поток.

Примером многопоточного процесса может служить работа приложения MS Word, где параллельно могут выполняться несколько потоков: один занимается вводом данных, другой выстраивает строки и страницы, третий проверяет орфографию и т.д.

Приоритеты потоков.

Процессы могут иметь следующие виды приоритетов:

Реального времени (Real time) – самый высокий приоритет (больший приоритетов многих задач самой операционной системы). Задержка в таком потоке полностью "вешает" ОС. Используется для обработки высокоскоростных потоков данных.

Высокий (High) – используется для задач, гарантированно завершающихся за короткое время (например опрос внешнего устройства). Применяется крайне редко.

Нормальный (Normal) – используется в подавляющем большинстве случаев. Не "подвешивает" работу других процессов.

Фоновый (Idle) – процесс получает кванты времени, если в очереди диспетчера задач нет потоков с более высоким приоритетом (например хранитель экрана).

Процесс можно запустить, в частности, программно вызовом функции WinExec или CreateProcess, с параметрами: имя приложения, параметры командной строки, вид окна приложения, приоритет приложения.

Приоритет потока, создаваемого в некотором процессе, может отличаться от приоритета процесса на плюс-минус два пункта (приоритет фонового потока 4 пункта, реального времени – 24 пункта).

Программирование потоков.

С точки зрения операционной системы поток – это объект. Для работы с ним в Delphi имеется соответствующий класс **TThread**. Объект этого класса создается до реального возникновения потока в системе и уничтожается после его исчезновения.

Рассмотрим основные методы данного класса.

`constructor` **`Create`**`(CreateSuspended :Boolean)` – создает объект (поток). Если параметр равен `False`, то поток будет запущен немедленно, а если `True`, то поток не выполняется, пока не буде вызван метод **Resume.**

`destructor` **`Destroy`** – завершает поток и освобождает связанные с ним ресурсы.

`procedure` **`Suspend`** – приостанавливает поток.

`procedure` **`Resume`** – запускает приостановленный поток.

`property` **`Priority`** `: integer` – представляет приоритет потока. Возможные варианты представлены в таблице:

Константа приоритета потока	Тип приоритета
`tpIdle`	фоновый
`tpLowest`	на 2 ниже приоритета основного процесса
`tpLower`	на 1 ниже приоритета основного процесса
`tpNormal`	равен приоритету основного процесса
`tpHigher`	на 1 выше приоритета основного процесса
`tpHighest`	на 2 выше приоритета основного процесса
`tpTimeCritical`	наивысший

`procedure` **`Synchronize`** `(Method : THreadMethod)` – используется для корректного обращения к компонентам VCL (визуальным компонентам Delphi) внутри потока. Обращение оформляется в виде метода без параметров, например:

```
procedure TForm1.SyncMessage;
begin
  ShowMessage(...);
end
```

который вызывается в потоке в качестве параметра процедуры **`Synchronize`** **`(`**`и никак иначе`**`)`**`:`

```
...
Synchronize(TForm1.SyncMessage);
...
```

`procedure` **`Execute; virtual; abstract;`** – главный метод потока. Переопределяя его в теле этого метода записывают тот код, который должен выполняться в рамках этого потока. Вызывается автоматически в конструкторе при создании потока (если указан параметр `False`) или методом **`Resume`**. Самим вызывать **`Execute`** не нужно!

Пример многопоточного приложения.

Рассмотрим приложение, в котором метка Label1 будет отображать текущее время. Это можно организовать с помощью стандартного компонента Ttimer, однако изменение содержимого метки оформим в виде потока.

Для этого добавим о основной модуль (Unit1) описание нового класса

```
type
  TMyTimer = class(TThread)
    procedure UpdateCaption;
    procedure Execute; override;
  end;
```

А в раздел `implementation` внесем тексты указанных процедур класса

```
procedure TMyTimer.UpdateCaption;
begin
  Form1.Label1.Caption := TimeToStr(Now);
end;

procedure TMyTimer.Execute;
begin
   repeat
     Synchronize(UpdateCaption);
   until terminated;
end;
```

Для работы с потоком в разделе var модуля опишем переменную, представляющую объект, связанный с потоком:

```
var
  MyTimer : TMyTimer;
```

Создание объекта (и, соответственно, запуск потока)Для работы с потоком опишем в обработчике события OnCreate формы:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  MyTimer:=TMyTimer.Create(False);
end;
```

Приложение готово, можно запускать.