МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 _____ 20___ г.

ОРГАНИЗАЦИЯ ПОТОКОВ В ПОИСКОВО-ПЕРЕБОРНЫХ ЗАДАЧАХ:

методические указания к лабораторным занятиям для магистров направления подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Курск 20__

УДК 004

Составитель: Е.А. Титенко

Рецензент
Кандидат технических наук *А. В. Киселев*

Организация потоков в поисково-переборных задачах: методические указания к лабораторным занятиям / Минобрнауки России, Юго-Зап. гос. ун-т; сост.: Е.А. Титенко. - Курск, 2022. - 12 с. - Библиогр.: с. 13.

Приводится теоретический материал и прикладной материал по классу TThread, рассмотрены особенности использования класса TThread для организации параллельных вычислений.

Методические рекомендации предназначены для студентов, обучающихся по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем очной и заочной форм обучения.

Текст печатается в авторской редакции.

Подписано в печать . Формат 60x84 1/16.
Усл.печ. л. 0,70 п.л . Уч.-изд. л. 0,63. Тираж 100 экз. Заказ.
Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель практической работы:

Изучение основных принципов организации параллельных вычислений на Delphi (изучение класса TThread) на поисково-переборных алгоритмах, определение количественных характеристик параллельного поиска (затраты памяти, оценка времени).

1 Основные понятия

Проблема организации поиска в условиях априорной неопределенности шагов имеет прежде всего ресурсное ограничение. Данное ограничение связано с наличием фиксированного количества активных исполнительных устройств (процессов), способных параллельно выполняться в вычислительной системе. Если количество активных устройств (процессов) превышает текущий коэффициент ветвления, то все ветви графа могут выполняться независимо с формированием различных промежуточных результатов. В противном случае, возникает конфликт за общий ресурс (активные устройства или процеccs) и необходимость выбора приоритетных из них. В рамках объектно-ориентированной операционной системы возможен механизм организации квазипараллельных вычислений, при котором с точки зрения программиста организуется параллельный обход графа с использованием потоков. Техническая поддержка параллельного программирования многопроцессорной организации позволит перейти от квазипараллельных вычислений к параллельным вычислениям в условиях априорной неопределенности.

1.1 Потоки в Delphi

Поток – это объект операционной системы (ОС), который представляет путь выполнения программы внутри отдельного процесса. Каждое приложение Win32 имеет по крайней мере один поток, часто называемый *основным*, или *стандартным*, но приложения для выполнения других задач вольны создавать дополнительные потоки. Delphi инкапсулирует потоковый объект API в объекте Object Pascal, именуемом **TThread**.

1.2 Основы работы класса TThread

Определение объекта TThread находится в модуле Classes и имеет следующий вид:

```
TThreadPriority = (tpidlj, tpLowest, tpLower, tpNormal, tpHigher,
txHighest, tpTimeCritical) ;
TThread = class
private
FHandle: THandle;
FThreadID: THandle;
WTerminated: Boolean;
FSuspended: Boolean;
FReturnValue: Integer;
function GetPriority: TThreadPriority;
procedure SetPriority (Value: TThreadPriority);
procedure SetSuspeuded (Value: Boolean);
```

```
protected
property ReturnValue: Integer read FReturnValue write
FReturnValue;
property Terminated: Boolean read FTerminated;
procedure Synchronize (Method: TThreadMethod);
procedure Execute; virtual; abstract;
public
constructor Create (CreateSuspended: Boolean);
destructor Destroy; override;
procedure Resume;
procedure Suspend;
procedure Terminate;
function WaitFor: Integer;
Property-Handle: THandle read FHandle;
property ThreadID: THandle read FThreadID;
property Priority: TThreadPriority read GetPriority write
SetPriority;
property Suspended: Boolean read FSuspended write
SetSuspended;
end;
```

Procedure Execute; virtual; abstract;

Одна из особенностей, которая отличает класс TThread от других полезных классов — это то, что вы для его использования должны породить дочерний класс. Причина, по которой нужно порождать дочерний класс в том, что метод Execute класса TThread объявлен как абстрактный.

Переопределяя метод Execute, мы можем тем самым закладывать в новый потоковый класс то, что будет выполняться при его запуске. Если поток был создан с аргументом CreateSuspended, равным False, то метод Execute выполняется немедленно, в противном случае Execute выполняется после вызова метода Resume.

constructor Create (CreateSuspended: Boolean);

Как и для любого другого класса, здесь нужен конструктор. В случае TThread он называется Create: в качестве аргумента он получает значение CreateSuspended. Если CreateGuspended равно True, вновь созданный поток не начинает выполняться до тех пор, пока не будет сделан вызов метода Resume. В случае если CreateSuspended имеет значение False, поток начинает исполнение и конструктор возвращает управление.

Деструктор Destroy вызывается, когда необходимость в созданном потоке отпадает. Деструктор высвобождает все ресурсы, связанные с объектом TThread.

Procedure Resume;

Метод Resume класса TThread вызывается, когда поток возобновляется после остановки или если он был создан с параметром CreateSuspended, равным True-Procedure Suspend;

Вызов метода Suspend приостанавливает поток с возможностью повторного запуска впоследствии. Метод Suspend приостанавливает поток вне зависимости от кода, исполняемого потоком в данный момент; аыполнбнзе продолжается с точки останова.

Function Terminate: Integer;

Для окончательного завершения потока (без последующего запуска) существует метод Terminate; он останавливает поток и возвращает

управление вызвавшему процессу только после того, как это произошло. Значение, возвращаемое функцией Terminate, соответствует состоянию потока. Примерами возможных состояний являются случай нормального завершения и случай, когда к моменту вызова Terminate поток уже завершился сам.

Function WaitFor: Integer;

Метод WaitFor позволяет одному потоку дождаться момента, когда завершится другой поток. Он возвращает код завершения потока.

Property Handle: THandle read FHandle; Property ThreadID: THandle read FThreadID;

Свойства Handle и ThreadID дают программисту непосредственный доступ к потоку средствами API Win32. Если разработчик хочет обратиться к потоку и управлять им, минуя возможности класса TThread, значения Handle и ThreadID могут быть использованы в качестве аргументов функций Win32 API. Например, если программист хочет перед продолжением выполнения приложения дождаться завершения сразу нескольких потоков, он должен вызвать функцию API WaitForMultipleObjects; для ее вызова необходим массив описателей потоков.

property Priority: TThrea^Priority;

Свойство Priority позволят запросить и установить приоритет потоков. Приоритет определяет, насколько часто поток получает время процессора. Естественно, программист захочет выделить главному потоку в приложении большее время, а потоку, например, с фоновой проверкой орфографии — меньшее. Допустимыми значениями приоритета являются tpidle, tpLowest, tpLower, tpNorrral, tpHigher, tpHighest и tpTimeCritical.

Property Suspended: Boolean ;

Свойство Suspended позволяет программисту определить, не приостановлен ли поток. С помощью этого свойства можно также запускать и останавливать поток. Установив Suspended в True, вы получите тот же результат, что и при вызове метода Suspend — приостановку. Наоборот, установка Suspended в False возобновляет выполнение потока, как и вызов Resume.

Для полноценного использования класса TThread нужно иметь некоторое представление и о методах секции protected. Описание защищенного метода выглядит следующим образом:

procedure Synchronize (Method: TThreadMethod);

Delphi предоставляет программисту метод Synchronize для безопасного вызова методов VCL внутри потоков. Во избежание ситуаций гонок, метод Synchronize дает гарантию, что к каждому объекту VCL одновременно имеет доступ только один поток. Аргумент, передаваемый в метод Synchronize — это имя метода, который производит обращение к VCL; вызов Synchronize с этим параметром — это то же, что и вызов самого метода. Такой метод не должен иметь никаких параметров и не возвращать никаких значений.

Property ReturnValue: Integer;

Свойство ReturnValue позволяет узнать и установить значение, возвращаемое потоком по его завершении. Эта величина полностью определяется пользователем. По умолчанию поток возвращает ноль, но если программист захочет вернуть другую величину, то простая переустановка ReturnValue внутри потока позволит получить эту информацию другим потокам. Это, к примеру, может пригодиться, если внутри пото-

ка возникли проблемы или с помощью ReturnValue нужно вернуть число не прошедших орфографическую проверку слов.

Property Terminated: Boolean;

Свойство Terminated позволяет узнать, произошел ли уже вызов метода Terminate или нет.

2 Практический пример создания многопотокового приложения в Delphi

Для того чтобы в Delphi создать экземпляр класса TThread, необходимо открыть меню File и выберать пункт New. Затем в появившемся диалоговом окне New Items необходимо выбрать объект типа поток (Thread Object). После этого в диалоговом окне New Thread Object необходимо ввести имя потока (например, TSimpleThread). Delphi создаст новый модуль, содержащий шаблон для нового потока. Вот как будет выглядеть его объявление:

```
type
  TSimpleThread = class(TThread)
        private
              { Private declarations }
        protected
              procedure Execute; override;
end;
```

Для создания функционального потомка класса TThread необходимо переопределить метод Execute():

```
procedure TSimpleThread.Execute;
begin
  {Код потока помещается здесь}
end;
```

Выполнить этот пример потока можно вызовом конструктора Create():

```
procedure TForm1.Button1Click(Sender:TObject);
var NewThread:TSimpleThread;
begin
  NewThread:= TSimpleThread .Create(False);
end;
```

3. Варианты заданий

Система правил					Кол-во целевых сост.
N	Кол-во правил	Тек. коэф. ветвления	Кол-во сост. сужения	Тек. коэф. сужения	
1	15	2	0	0	1
2	20	1-3	1	2	2
3	15	2-3	3	3	3
4	20	2	0	0	4
5	15	1-3	2	2	4
6	20	2-3	3	3	3
7	15	2	0	0	2
8	20	1-3	1	2	1
9	15	2-3	0	0	2
10	20	2	2	2	3

11	15	1-3	1	3	4
12	20	2-3	2	0	3
13	15	2	0	0	2
14	20	1-3	2	2	1
15	15	2-3	1	2	2
16	20	2	3	0	3
17	15	1-3	2	3	4
18	20	2-3	3	0	3
19	15	2	0	0	2
20	20	1-3	1	2	12
21	15	2-3	3	0	3
22	20	2	1	3	3
23	15	1-3	2	0	2
24	15	1-3	0	1	1
25	20	2-3	3	2	2
26	15	2	2	0	3
27	20	2	1	1	4
28	15	1-3	0	2	3
29	20	2-3	2	0	2
30	15	2	0	0	1
31	20	1-3	2	1	2
32	15	1-3	2	2	3
33	20	2-3	3	1	4
34	15	2	2	0	3
35	20	2-3	0	2	2

где A – обход от данных, B – обход от цели.

Во всех вариантах правила задаются со структурой *→*, где *-состояние, обозначаемое буквой латинского алфавита. Среди состояний выделяется единственное начальное (буква A) и заданные пользователем целевые состояния.

4 Контрольные вопросы

1. Что такое потоки? Для чего используются потоки?
2. Для чего предназначен класс TThread в Delphi?
3. Определение объекта TThread.
4. Для чего предназначен метод Execute класса TThread?
5. Особенности конструктора класса TThread.
6. Приостановка и возобновление потоков.
7. Завершение потоков. Ожидание потоков.
8. Свойства Handle и ThreadID.
9. Свойство ReturnValue. Метод Synchronize.
10. Свойство Terminated.

Библиографический список

1. Интеллектуальные системы [Электронный ресурс] : учебное пособие; Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Оренбургский государственный университет». - Оренбург : Издательство ФГБОУ ВПО «ОГУ», 2013. - 236 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=259148.
2. Потапов А.С. Технологии искусственного интеллекта [Электронный ресурс]: учебное пособие. - СПб: СПбГУ ИТМО, 2010. - 218 с.
3. Серегин, М. Ю. Интеллектуальные информационные системы [Электронный ресурс]: учебное пособие / М. Ю.Серегин , М. А Ивановский, А. В.Яковлев ; Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». - Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2012. - 205 с.
4. Сидоркина, И. Г. Системы искусственного интеллекта [Текст] : учебное пособие / И. Г. Сидоркина. - Москва : КНОРУС, 2016. - 246 с.
5. Андрейчиков, А. В. Интеллектуальные информационные системы [Текст] : учебник / А. В. Андрейчиков, О. Н. Андрейчикова. - М. : Финансы и статистика, 2006. - 424 с.