А.В. Макаров, С.Ю. Скоробогатов, А.М. Чеповский

Учебный курс “CIL и системное программирование в Microsoft .NET”

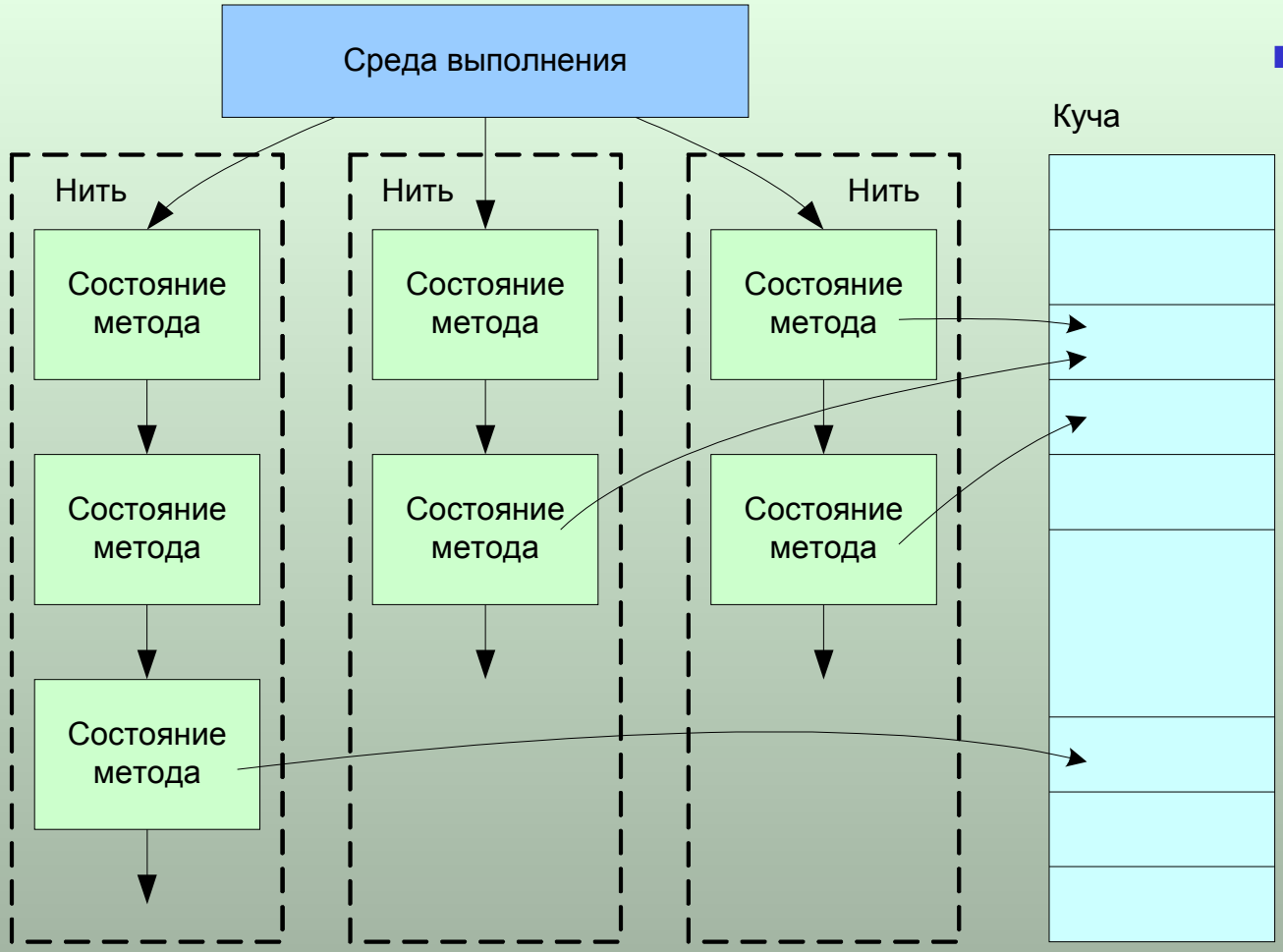
Лекция 3.

Виртуальная система выполнения

Введение

- Виртуальная система выполнения (Virtual Execution System – VES) представляет собой абстрактную виртуальную машину, способную выполнять управляемый код
- Предназначение VES – служить образцом, которому должна соответствовать любая реализация CLI. Какую бы технологию не использовала эта реализация для выполнения программ, эта технология должна работать так же, как работала бы виртуальная система выполнения
- VES является значительно более абстрактной моделью, чем виртуальная машина Java (Java Virtual Machine – JVM)

3.1. Состояние виртуальной машины



- Состояние виртуальной машины является совокупностью состояний нитей и состояния кучи

Состояние нити

- Состояние нити представляет собой односвязный список состояний методов. Метод, состояние которого находится в самом конце этого списка, является активным. Если активный метод вызовет другой метод, то в конец списка будет добавлено новое состояние для вызываемого метода. Если же активный метод закончит свою работу, то его состояние будет удалено из списка
- Такая схема состояния нити является абстракцией традиционной модели последовательности вызовов методов

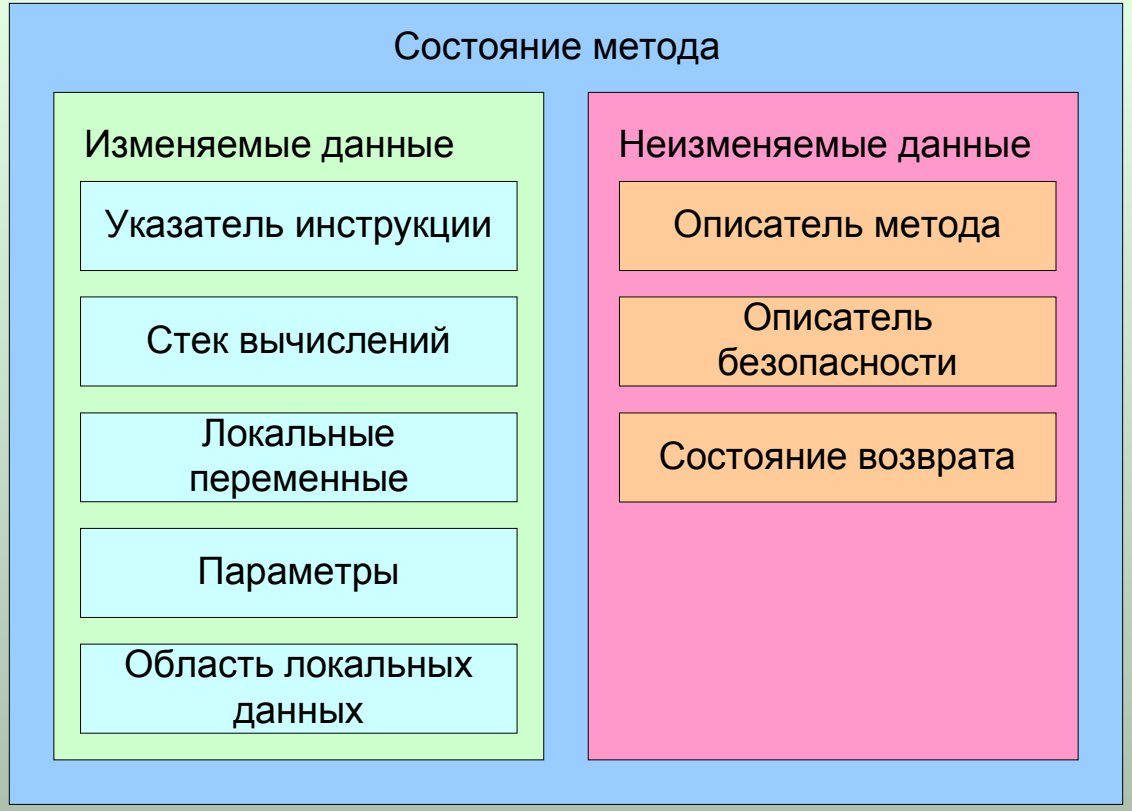
Состояние кучи

- Состояние кучи определяется состояниями содержащихся в ней объектов
- Допустимо существование сразу нескольких куч
- Спецификация VES предполагает, что для удаления ненужных объектов из кучи будет использоваться какой-нибудь алгоритм автоматического управления памятью. При этом детали этого алгоритма не рассматриваются, то есть в реализациях CLI могут применяться различные алгоритмы сборки мусора

Адресное пространство и побочные эффекты

- Состояния нитей и состояние кучи должны находиться в общем адресном пространстве
- Параметры и локальные переменные метода являются частью состояния метода и, следовательно, видимы только для нити, в которой этот метод выполняется
- Параметры и локальные переменные могут содержать ссылки на объекты, размещенные в куче и видимые для других нитей. Поэтому изменение объекта в куче из одной нити может повлиять на работу других нитей и считается побочным эффектом

3.2. Состояние метода



- Изменяемые данные доступны из тела метода для чтения и записи, в то время как неизменяемые данные либо доступны только для чтения, либо вообще предназначены для внутреннего использования в системе выполнения

Изменяемые элементы состояния метода

- Указатель инструкции (Instruction Pointer).

  Адрес следующей инструкции в теле метода
- Стек вычислений (Evaluation Stack).

  Каждое состояние метода имеет собственный стек вычислений, содержимое которого сохраняется при вызове методов
- Локальные переменные (Local Variable Array).

  Для хранения локальных переменных в состоянии метода предусмотрена отдельная область памяти, состоящая из так называемых слотов (slots). Каждой локальной переменной соответствует свой слот
- Параметры (Argument Array).

  Фактические параметры записываются в специальную область памяти, которая организована так же, как и область локальных переменных.
- Область локальных данных (Local Memory Pool).

  В языке CIL предусмотрена инструкция **localloc**, позволяющая динамически размещать объекты в области памяти, локальной для метода

Неизменяемые элементы состояния метода

- Описатель метода (methodInfo handle).

  Содержит сигнатуру метода, в которую входят количество и типы формальных параметров, а также тип возвращаемого значения. Кроме этого, описатель метода включает в себя информацию о количестве и типах локальных переменных и об обработчиках исключений.

  Описатель метода доступен из кода метода, но в основном он используется системой выполнения при сборке мусора и обработке исключений.

- Описатель безопасности (Security Descriptor).

  Используется системой безопасности CLI и недоступен из кода метода.

- Состояние возврата (Return State Handle).

  Служит для организации списка состояний методов внутри системы выполнения и недоступно из кода метода. Фактически представляет собой указатель на состояние метода, из тела которого был вызван текущий метод

3.2.1. Стек вычислений

- Не смотря на то, что большинство современных процессоров для организации вычислений используют регистры, в виртуальной системе выполнения вместо регистров применяется стек вычислений
- Стек вычислений в VES состоит из слотов
- Глубина стека всегда ограничена и задается статически в заголовке метода
- На входе метода стек вычислений всегда пуст
- Слоты стека вычислений не адресуются, то есть мы не можем получить указатель на какой-нибудь слот и записать в него значение

Типы данных на стеке вычислений

- Каждый слот стека вычислений может содержать ровно одно значение одного из следующих типов:
  - int64 – 8-байтовое целое со знаком
  - int32 – 4-байтовое целое со знаком
  - native int – знаковое целое, разрядность которого зависит от аппаратной платформы (может быть 4 или 8 байт)
  - F – число с плавающей точкой, разрядность которого зависит от аппаратной платформы (не может быть меньше 8 байт)
  - & – управляемый указатель
  - O – объектная ссылка
  - Пользовательский тип-значение
- Слоты стека вычислений могут иметь различный размер в зависимости от типов записанных в них значений

Работа с типами данных, не поддерживаемыми напрямую стеком вычислений

- Короткие целые типы (bool, char, int8, int16, unsigned int8, unsigned int16) при загрузке на стек вычислений расширяются до int32
- unsigned int32 превращается в int32, unsigned int64 превращается в int64, unsigned native int превращается в native int
- Типы float32 и float64 при копировании на стек вычислений преобразуются к типу F
- Типы-перечисления при копировании на стек вычислений автоматически превращаются в целые типы
- Любой управляемый указатель считается имеющим тип &, а любая объектная ссылка представляется типом O

3.2.2. Локальные переменные и параметры

- Для хранения локальных переменных и параметров метода используются два массива, которые состоят из слотов. При этом каждой переменной и каждому параметру соответствует ровно один слот
- Для доступа к локальным переменным и параметрам используются их индексы в массивах переменных и параметров. При этом нумерация осуществляется с нуля
- Слоты, из которых состоят массивы переменных и параметров, адресуемы

3.2.4. Область локальных данных

- Область локальных данных является составной частью состояния метода и используется для размещения объектов, тип и/или размер которых неизвестен на этапе компиляции, но которые по тем или иным причинам нежелательно размещать в куче
- Область локальных данных существует ровно столько, сколько исполняется метод, состоянию которого она принадлежит. После прекращения работы метода она автоматически освобождается
- В верифицированном коде использование области локальных данных запрещено