Сопроцедуры в событийно-ориентированных архитектурах

Д. В. Луцив

Кафедра системого программирования СПбГУ

25 февраля 2010

Содержание

Примеры

- Телекоммуникационное ПО
  - ПО телефонной связи
  - Стек почти любых протоколов
- Десктопное ПО

Архитектура

- Цикл обработки событий
- Очередь событий
  - События м.б. из внешнего мира или внутренние
- Обработчики событий
- Состояния (явные или нет)

Проблемы

- Фоновые задачи
  - Инициируются обработчиком события
  - Выполняются длительное время
  - Во время выполнения не обрабатывают события
  - При этом не должны ухудшать «реактивность»
- Некоторые обработчики событий могут работать немного дольше, не являясь фоновыми задачами, но ухудшая «реактивность».

Проблемы

- Фоновые задачи
  - Инициируются обработчиком события
  - Выполняются длительное время
  - Во время выполнения не обрабатывают события
  - При этом не должны ухудшать «реактивность»
- Некоторые обработчики событий могут работать немного дольше, не являясь фоновыми задачами, но ухудшая «реактивность».

Вытесняющая многозадачность

Процессы и потоки. Хороши универсальностью.
Плохи:

- малой детерменированностью
- ресурсоёмкостью

Явный ввод состояний вручную

Плох:

- подходит для ПО с описанными состояниями
- количество состояний растёт экспоненциально

Кооперативная многозадачность

- подходит только для заранее спроектированного ПО
- минимум накладных расходов

Вытесняющая многозадачность

Процессы и потоки. Хороши универсальностью.
Плохи:

- малой детерменированностью
- ресурсоёмкостью

Явный ввод состояний вручную

Плох:

- подходит для ПО с описанными состояниями
- количество состояний растёт экспоненциально

Кооперативная многозадачность

- подходит только для заранее спроектированного ПО
- минимум накладных расходов

Использование кооперативной многозадачности

Twisted Matrix

- Общий цикл обработки событий
- Всё взаимодействие со средой через объекты подклассов `Deferred`
- Для обработки любых ситуаций — callback в данные объекты

БАСК АН

Закрытая десктопная система (система поддержки рабочего места оператора), обеспечивающая интерфейс к устройствам БАСК. Наряду с интерфейсными задачами выполняет обмен данными с БАСК в фоновом режиме.

Пример Twisted Matrix

```
from twisted.web.client import getPage
from twisted.internet import reactor

def printContents(contents):
    '''This is the 'callback' function'''
    print "Deferred has called back with:"
    print contents
    # Stop the Twisted event handling system.
    reactor.stop()

# call getPage,
deferred = getPage('http://twistedmatrix.com/')
# add a callback to the deferred
deferred.addCallback(printContents)
# start the Twisted event handling system
reactor.run()
```

Сопроцедура (сопрограмма)

- Много точек входа
- Много точек возврата управления
  - Возможно, с возвратом значения
- Механизм сохранения состояния от возврата до следующего входа

Сопроцедуры Си (I)

```
#include <stdio.h>
#include <setjmp.h>
typedef struct {
  jmp_buf coroutine_buf; int coroutine_done;
  char *cname; int i;
} coroutine_struct;

void coroutine(coroutine_struct *cs) {
  cs->coroutine_done = 0;
  for(cs->i = 0; cs->i < 3; ++cs->i) {
    printf("coroutine %s: i=%d\n", cs->cname, cs
        ->i);
    if(!setjmp(cs->coroutine_buf))
      return;
  }
  cs->coroutine_done = 1;
}
```

Сопроцедуры Си (II)

```
int main() {
  char j;
  coroutine_struct cs1, cs2;
  cs1.cname = "c1"; cs2.cname = "c2";

  coroutine(&cs1); coroutine(&cs2);
  for(j = 'a'; j < 'd'; ++j) {
    printf("main, j=%c\n", j);

    if(!cs1.coroutine_done)
      longjmp(cs1.coroutine_buf, 0);

    if(!cs2.coroutine_done)
      longjmp(cs2.coroutine_buf, 0);
  }
  return 0;
}
```

Сопроцедуры C#: описание

```
using System;
using System.Collections.Generic;
public class CoRoTest
{
  public static IEnumerable<int> GetIntegers(int
      upper)
  {
    for(int x = 0; x < upper; ++x)
      yield return x;
  }

  public static void Main()
  {
    foreach(int z in GetIntegers(5))
      Console.WriteLine(z);
  }
}
```

Сопроцедуры C#: реализация (I)

```
[CompilerGenerated]
private sealed class <GetIntegers>c__Iterator0 :
IEnumerable<int>, IEnumerator<int>, IEnumerator,
IDisposable, IEnumerable
{
    // Fields
    internal int $current;
    internal int $PC;
    internal int <$>upper;
    internal int <x>__0;
    internal int upper;

    // Methods
    // ...
    public bool MoveNext();
    // ...
}
```

Сопроцедуры C#: реализация (II)

```
public bool MoveNext()
{
  uint num = (uint) this.$PC;
  this.$PC = -1;
  switch (num)
  {
    case 0:
      this.<x>__0 = 0;
      break;
    case 1:
      this.<x>__0++;
      break;
    default:
      goto Label_006B;
  }
  if (this.<x>__0 < this.upper
      )
  {
    this.$current = this.<x>
        __0;
    this.$PC = 1;
    return true;
  }
  this.$PC = -1;
Label_006B:
  return false;
}
```

Haskell (забавный пример)

- Любая функция — сопроцедура, т.к. язык ленивый
- Это нам ничего не даст, т.к. у функций нет побочных эффектов

Реализовать сопроцедуры, общающиеся с внешним миром можно только с помощью монады IO:

$$(>>=) : IO\ x \times (x \to IO\ y) \to IO\ y,$$

а на самом деле даже

$$(>>=) : IO\ x \to (x \to IO\ y) \to IO\ y.$$

Синтаксический сахар «do»

```
main = do
 sa <- getLine
 print sa
```

Оператор >>=, \ ≡ λ

```
main =
 getLine >>=
 \ sa -> print sa
```

Haskell (забавный пример)

- Любая функция — сопроцедура, т.к. язык ленивый
- Это нам ничего не даст, т.к. у функций нет побочных эффектов

Реализовать сопроцедуры, общающиеся с внешним миром можно только с помощью монады IO:

$$(>>=) : IO\ x \times (x \to IO\ y) \to IO\ y,$$

а на самом деле даже

$$(>>=) : IO\ x \to (x \to IO\ y) \to IO\ y.$$

Синтаксический сахар «do»

```
main = do
 sa <- getLine
 print sa
```

Оператор >>=, \ ≡ λ

```
main =
 getLine >>=
 \ sa -> print sa
```

«Зелёные» потоки

Для пользователя — вытесняющая многозадачность, для интерпретатора или ВМ — кооперативная.

- Использовались в ранних (до 1.2) версии Java.
- До сих пор используются современные версии Python и некоторых других интерпретаторов.

Плюсы

- Реализация не зависит от окружения.
- Не прерывается интерпретация отдельных команд виртуальной машины.

Минус

Не используются возможности современной аппаратуры для организации параллельных вычислений.

«Зелёные» потоки

Для пользователя — вытесняющая многозадачность, для интерпретатора или ВМ — кооперативная.

- Использовались в ранних (до 1.2) версии Java.
- До сих пор используются современные версии Python и некоторых других интерпретаторов.

Плюсы

- Реализация не зависит от окружения.
- Не прерывается интерпретация отдельных команд виртуальной машины.

Минус

Не используются возможности современной аппаратуры для организации параллельных вычислений.

Поддержка в разных языках

Языки со встроенной и/или легко реализуемой поддержкой сопроцедур:

- C#
- Io
- Lua
- Modula-2
- Python
- Ruby
- Simula
- Perl 6
- AngelScript
- Forth

- Любой обработчик события — сопроцедура.
- После возврата управления, если сопроцедура не завершилась, она (вместе со старым контекстом, разумеется) опять добавляется в очередь диспетчера событий.
- Точки возврата управления определяются прикладным программистом (не принципиально).

- Через вторжение на стек (вариант Си).
- При помощи компилятора АЯВУ (вариант C#), или, для RTST, при помощи генератора SDL $\longrightarrow$ АЯВУ.
- При помощи виртуальной машины (как в Python, Ruby и т.д.)

Спасибо