Скриптовые языки программирования

Доклад на семинаре по специальности

Студент гр. 4057/2 Руцкий Владимир

20.10.2009

Содержание

- Введение
- Особенности скриптовых языков программирования
- Типы скриптов и примеры
- Заключение

Введение

Иерархия языков программирования

- Машинные коды, язык ассемблера.
  - Работа с отдельными ячейками памяти, регистрами
- Компилируемые в машинные коды (C, C++).
  - Работа с примитивными типами данных (числа, массивы, структуры/классы)
- Компилируемые в байт-код или интерпретируемые (Java, C#, Python).
  - Более сложные типы данных и более сложные элементарные операции над ними
- Работающие со сложными типами данных (с таблицами, программами — SQL, shell)

Сложность элементарных операций

<здесь будет приведена диаграмма количества машинных команд необходимых для выполнения каждой элементарной операции в соответствующем классе языка программирования с предыдущего слайда>

Сложность написания программ

<здесь будет приведена статистика, сколько строк кода (и сколько времени) занимает решение какой-то определённой задачи, в различных классах языков программирования с предпредыдущего слайда>

Эволюция языков программирования

Чем выше уровень языка, тем

- Проще и быстрее разработка программы
- Медленнее работа программы

Вычислительная техника развивается столь стремительно, что применение сильно высокоуровневых языков программирования является оправданным во многих задачах.

Появление скриптов

- 1960-е годы. Shell-скрипты. Автоматизация работы человека-оператора по вводу команд ОС в системах с разделением времени
- «Скриптовый язык», «язык сценариев» - язык записи «сценария», последовательности выполнения команд ОС

Особенности скриптовых языков программирования

Назначение скриптовых языков программирования

Быстрое и простое связывание и управление готовыми объектами (функциями, программами)

- Для создания новых программ на основе существующих
- Для автоматизации различных рутинных операций
- Для быстрой разработки технологических тестов
- Для задания сценариев работы программы не программистами
- Для управления специальными данными

Типы данных

- Бестиповые — более абстрактный и универсальный код
- Универсальные типы — возможность произвольного связывания различных компонент

Недостаток:

- Нетипизированность данных не позволяет выявить ошибочное использование переменных до начала выполнения скрипта

Особенности среды выполнения скриптов

Выполнение скриптов:

- Практически никогда не компилируются в машинные коды
- Компилируются в байт-код (обычно лишь для оптимизации скорости выполнения, кешируются)
- Чаще всего интерпретируются «на лету»

Многие скрипты — кроссплатформенные

Особенности синтаксиса

- Чаще всего позволяет «построчное» выполнение кода — даёт возможность программирования «на лету», прямо во время выполнения программы
- Минимализм в конструкциях языка

Типы скриптов и примеры

Классификация типов языков

- <классификация>
- <языки, которые будут рассмотрены>

Управление последовательностью команд ОС

- «Shell» — «оболочка» — интерфейс между функциями ядра/системы и конечным пользователем
  - GUI — graphical user interface
  - CLI — command-line interface
- Первые оболочки ОС — текстовые (CLI)
- Первая командная оболочка — 1963 г., MIT, для ОС с разделением времени

История Shell

- CTSS OS (Compatible Time-Sharing System) — одна из первых ОС с разделением времени (1961, MIT Computation Center)
- Louis Pouzin (родился в 1931 во Франции) участвовал в проектировании CTSS, разработал программу RUNCOM (1963/64), позволяющую выполнять команды ОС в текущей директории. Первым ввёл термин «shell»

Развитие Shell

<TODO: пока не знаю точно о чем это>

Четыре поколения shell:

1. Thomson shell, Mashey shell
2. C-shell, Bourne shell
3. tcsh, ksh88
4. ksh93, bash, zsh, Microsoft Power Shell

ОС Unix shell: sh

<TODO: Изменить заголовок слайда, мне не удаётся сделать это в OpenOffice сейчас :)>

- **Thompson shell** — первая оболочка для Unix, разработал Ken Thompson в в AT&T Bell Laboratories в 1971
- **sh** — *Bourne shell*, разработал Stephen Bourne в AT&T Bell Laboratories и был выпущен в 1977 как оболочка по умолчанию для Version 7 Unix. Заменил Thompson shell
- **bash** — *Bourne-again shell, p*азработал в 1987 Brian Fox в рамках проекта GNU.
- bash — обратно совместимое надмножество над sh
- bash до сих пор продолжает развиваться и широко используется в качестве shell по умолчанию в различных Unix-like ОС

Возможности bash

- Сейчас bash — полноценный язык программирования, по историческим причинам ориентированный на задание последовательности выполнения команд ОС
- «Стандартной библиотекой» для bash является набор стандартных утилит Unix.

Возможности bash в связке со стандартным набором программ Unix

<find, grep и некоторые примеры>

ОС MS Windows CLI: cmd.exe

- **COMMAND.COM** — shell по умолчанию для ОС DOS, и CLI ОС MS Windows 9x/Me <TODO: где и когда был разработан впервые>
- **CMD.EXE** — shell для ОС MS Windows выше Windows 2000, разработал Therese Stowell для MS Windows NT

Возможности cmd.exe

<TODO>

<+ Примеры>

Другие оболочки ОС

- <Другие Unix shells>
- <MS Power Shell>

Итог: современное использование оболочек ОС

<TODO>

Языки для обработки текстов

- Текст — цепочка символов — универсальный тип данных
- Алгоритмы, принимающие цепочку символов и возвращающие цепочку, можно связывать друг с другом в произвольном порядке

  <схема перенаправления вывода>
- Shell превосходно подходит для обработки цепочек символов: вход и выход программ — цепочки символов и shell имеет встроенную возможность для перенаправления вывода одной команды ОС на вход другой

awk

- <История создания, возможности, примеры>

Web-скрипты

- Чаще всего обмен информацией организован внутри пар клиент-сервер
- Общие для большинства схем взаимодействия клиент-сервер рутинные операции:
  - генерация данных на сервере,
  - подготовка к передаче клиенту,
  - отображение данных на стороне клиента,
  - приём данных от клиента

Серверные Web-скрипты

- <PHP, (ASP, Ruby on rails, SMX)>

Web-скрипты. Клиентские

- <JavaScript, VBScript>

Языки общего назначения

Perl. Описание

- **Perl** — Practical Extraction and Reporting Language — высокоуровневый язык общего назначения.
- Первоначально разработал Larry Wall в 1987 в США для обработки отчетов
- Предоставляет широкие возможности для обработки текстов
- Ориентирован на решение практических задач, нежели красоту/изысканность синтаксических конструкций
- Процедурный, с поддержкой объектно-ориентированной и функциональной парадигм программирования
- Автоматическое управление памятью. Сборщик мусора
- Интерпретируемый. В Perl 6 появится возможность компилирования в байт-код
- Специальные синтаксические конструкции для работы с регулярными выражениями и потоками ввода/вывода
- Девизы:
  - «There's more than one way to do it» (TMTOWTDI) — «Есть больше одного способа сделать это»
  - «Easy things should be easy and hard things should be possible» — «Простые вещи должны быть простыми, а сложные вещи — возможными»

Perl. Синтаксис

- Унаследовал общую структуру синтаксиса от языка Си
- Возможно создавать сложные структуры данных, как массивы/хеши ссылок на более простые типы
- Возможность создания модулей
- Примеры.

```
sub GCD {
  my ($a, $b) = @_;
  my $c = $b;
  while ($a > 0)
  {
    $c = $a;
    $a = $b % $a;
    $b = $c;
  }
  return $c;
}
print GCD(42, 56);
```

```
$number = 5;
$string = "String";
$multilined_string = <<EOF;
Multilined string,
terminating with the word "EOF".
EOF

@array = (1, 2, "three");
print $array[1]; # "2"

%hash = ('one' => 1, 'two' => 2);
print $hash{'one'}; # "1"

$ref = \$number;
print $ref; # SCALAR(0x14ef640)
print $$ref; # 5

$circle={
  center=>{x=>210, y=>297},
  radius=>53,
};
```

Perl. Особенности

- Встроенная в синтаксис поддержка регулярных выражений
  - $x =~ m/abc/; # Истина, если $x содержит «abc»
  - $x =~ s/abc/def/; # Заменит «abc» на «def» в $x
  - $x =~ m/(\d+)\.(\d+)\.(\d+)\.(\d+)\./ # Найдёт первое вхождение IP-адреса, и сохранит его части в переменных $1, $2, $3, $4
  - @a = ($x =~ m/(\d+)/g); # Сохранит в массив @a все найденные в строке числа
  - $x =~ s/(\d+) б /$1 Б/g # Заменит «X б» на «X Б»
- Регулярные выражения Perl удобны и включены во многие другие языки программирования
- Пример. Печать простых чисел:

  perl -wle '(1 x $_) !~ /^(11+)\1+$/ && print while ++ $_'
- Встроенные в синтаксис возможности для работы с потоками ввода/вывода

```
while ( <> ){ # построчное чтение из выбранного потока в переменную $_
  if ( /print/ ){ # если строка в $_ содержит «print» (регулярное выражение)
    print;     # печатаем содержимое переменной $_
  }
}
```

Perl. Применение

- Применение Perl:
  - Разработка инструментов системного администрирования
  - Обработка почты
  - CGI-сценарии
  - Обработка текстов
  - Работа с базами данных
  - Разработка средств тестирования
- Perl используют:
  - Amazon — онлайн сервисы
  - The BBC — обработка данных
  - Ebay — онлайн сервисы
  - Yahoo! — онлайн сервисы
  - NASA — скриптование в различных системах рассчета

Python. Описание

- **Python** — высокоуровневый язык общего назначения
- Первоначально разработал Guido van Rossum в 1990 в нидерландском CWI
- Акцент на производительность разработчика и читаемость кода
- Объектно-ориентированный, с поддержкой процедурной и функциональной парадигм
- Минималистичный синтаксис
- Динамическая типизация
- Автоматическое управление памятью. Сборщик мусора
- Интерпретируемый и/или компилируемый в байт-код
- Полная интроспекция
- Механизм обработки исключений
- Интерактивный режим выполнения

Python. Синтаксис

- Отступы определяют разделение программы на блоки
- Возможность написания классов
- Поддержка разделения на модули
- Переменные — имена ссылок на значения
- Коллекции (контейнеры):
  - Списки
  - Кортежи
  - Словари
  - Множества

  - Передаются по ссылке
- Примеры:

```
def GCD(a, b):
    while b > 0:
        a,b = b,a%b
    return a

print GCD(42, 56)
```

```
number = 5
complex_number = 1.5 + 0.5j
print complex_number.imag # "0.5"
big_number = 2**1000
print len(str(big_number)) # "302"

string = "String"
multilined_string = """
Multilined string,
Yep.
"""
unicode_string = u"Уникод"

list_var = [1, 2, 'three']
tuple_var = (1, 2, 'three')
print list_var[1] # "2"

dictionary = {'one': 1, 'two': 2}
print dictionary['one'] # "1"
set_var = set([1, 2, 'five'])
print 'five' in set_var # "True"
```

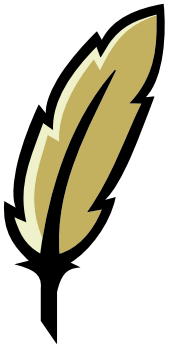
Python. Особенности

- Полная интроспекция
- Поддержка ООП
  - Абстракция — утиная типизация
  - Инкапсуляция — ограниченная ввиду интроспекции
  - Множественное наследование
  - Полиморфизм — все функции виртуальные
- Механизм обработки исключений
- Концепция итераторов
  - Единая схема итерирования
  - Лёгкое создание генераторов
  - Мощная библиотека для работы с итераторами
- Встроенная в синтаксис поддержка парадигм функционального программирования
  - Функции высшего класса
  - Лямбда-функции
  - List comprehension

Python. Применение

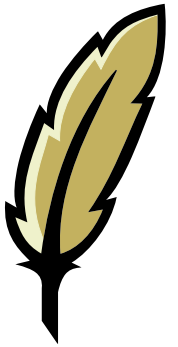
- Применение Python:
  - Часто используется как основной язык программирования
  - Быстрое прототипирование
  - Разработка инструментов для системного администрирования
  - Веб-программирование
  - Встроенный скриптовый язык
- Python используют:
  - Как встроенный скриптовый язык
    - 3D редакторы: Maya, MotionBuilder, Softimage, Blender
    - 2D редакторы: GIMP, Inkscape, Scribus, and Paint Shop Pro
    - ГИС решение ESRI ArcGIS
  - Скриптование в играх: Civilization IV, EVE Online
  - YouTube — для системного многраммирования
  - Для решения широкого спектра задач: Google, Yahoo!, CERN, NASA

Tcl. Описание



- **Tcl** — *Tool Command Language* — «тикль» — высокоуровневый язык общего назначения
- Первоначально разработал John Ousterhout в 1988 в Калифорнийском институте в Беркли, США
- Процедурный язык программирования с поддержкой
  - метапрограммирования,
  - функциональной парадигмы,
  - модели управления программой на основе событий
- Базовая реализация не содержит поддержки ООП, но есть альтернативные реализации, включающие её
- Ортогональность
- Компилируется в байт-код, а затем выполняется

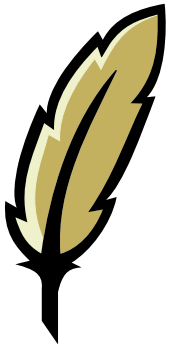
Tcl. Синтаксис



- Скрипт состоит из команд, разделённых переводом строки или точкой с запятой
- Команда:

  CommandName arg1 arg2 … argN
- Специальные символы:
  - $ — подстановка значения переменной
  - [] — подстановка результата выполнения команды внутри скобок.
  - "" — группировка аргументов в один с подстановкой значений переменных
  - {} — группировка аргументов в один без подстановки значений переменных.

Tcl. Конструкции языка



- Реализованы через общий синтаксис
- **set** a 5 # a = 5
- **expr** expression
  - **set** sum [**expr** $a + $b]
- **while** { cond } { expr }
  - **while** { $x < 10 } { **puts** $x; **incr** $x }
- **if** { cond } { expr }
- **proc** name { args } {

  expr (**return** expr)

  }

- **switch** options key {

  pattern1 { expr }

  pattern2 { expr }

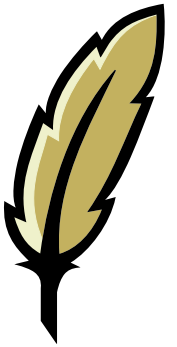
  …

  default { expr }

  }

- **foreach** iter values_list { expr }

Tcl. Типы данных

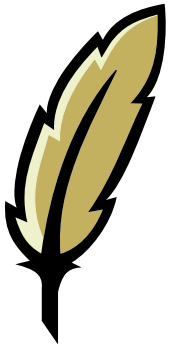


- Строки
- Списки строк
- Ассоциативный массив
- Примеры:

```
set number 5
set string "Some string with number $number"
set string_list {e1 e2 e3}
set map_var(one) 1
set map_var(two) 2
puts $map_var(two) # «2»
```

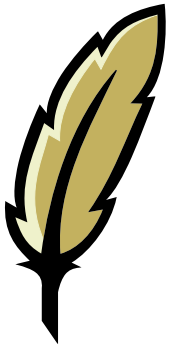
```
proc gcd { a b } {
  while { $b > 0 } {
    set t $b
    set b [expr $a % $b]
    set a $t
  }
  return $a
}

puts [gcd 42 56]
```

Tcl. Функциональное программирование



- Функциональная парадигма используется редко
- Возможность создания функций высшего порядка и абстракции функции

Tcl/Tk. Задание GUI (1/2)

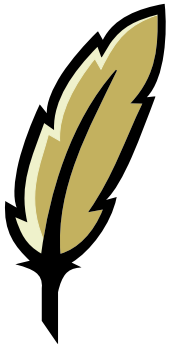


- Tk — Tool Kit — «инструментарий» — кроссплатформенная библиотека базовых элементов GUI
- Разработал John Ousterhout как расширение Tcl
- С использованием специальных библиотек может использоваться с другими языками программирования
- Представляет собой набор Tcl-команд для задания иерархии элементов GUI, и привязывания функций обработчиков к возникающим событиям

Tcl/Tk. Задание GUI (2/2)

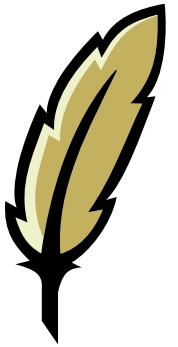
- Пример

Tcl. Применение (1/2)



- Быстрое прототипирование
- Создание графических интерфейсов (Tcl/Tk)
- Встроенный скриптовый язык
- Тестирование
- Написание хранимых процедур PostgreSQL (PL/Tcl)
- Иногда создание CGI-скриптов

Tcl. Применение (2/2)



- Продукты, использующие Tcl:
  -
- Компании, использующие Tcl:
  -

Ruby

Lua

Узконаправленные языки

- <MAXScript (3DSMax), MATLAB, R, Bison, консоль в некоторых играх (Quake, Half-Life), макросы MS Word/Excel, OpenOffice.>

Сравнение скриптовых языков программирования

<здесь будет сводная таблица характеристик различных скриптовых языков программирования, как Вы предложили>

Заключение

Скриптовые языки — набор удобных и надёжных инструментов для быстрого и простого решения широкого спектра задач

<+++>

Источники информации

<будет заполнено позже>

Благодарю за внимание!