

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 предыдущего 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)
 - Возможно выполнение JIT
- 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 — shell — 1963 г., MIT, для ОС с разделением времени

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 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. Заменял Thompson shell
- **bash** — *Bourne-again shell*, разработал 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. Общая характеристика



- **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 расчета

Python. Общая характеристика



- **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
- Полная интроспекция
- Механизм обработки исключений
- Интерпретируемый и/или компилируемый в байт-код
- Интерактивный режим выполнения

Python. Синтаксис



- Отступы определяют разделение программы на блоки
- Возможность написания классов
- Поддержка разделения на модули
- Переменные — имена ссылок на экземпляры классов
- Коллекции (контейнеры):
 - списки,
 - кортежи,
 - словари,
 - множества,
 - некоторые другие типы
- 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"  
multiline_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"
```

```
squares = [v ** 2 for v in range(0, 10) if random.randint(0, 1) == 1]  
for i, v in enumerate(squares):  
    print "%2d\t%2d" % (i, v)
```

```
# Вывод:
```

```
# 0    0  
# 1    1  
# 2    9  
# 3   16  
# 4   25  
# 5   49
```

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
- Декораторы
- Встроенное документирование кода

Python. Применение



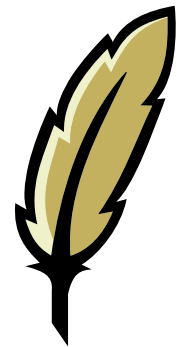
- Применение Python:
 - Часто используется как основной язык программирования
 - Быстрое прототипирование
 - Разработка инструментов для системного администрирования
 - Веб-программирование
 - Встроенный скриптовый язык
- Python используют:
 - Как встроенный скриптовый язык
 - 3D редакторы: Maya, MotionBuilder, 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 многопрограммирования
 - Для решения широкого спектра задач: Google, Yahoo!, CERN, NASA

Tcl. Общая характеристика



- **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. Особенности

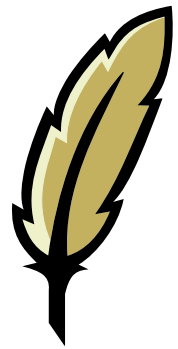


- Скрипт состоит из команд, разделённых переводом строки или точкой с запятой
- Команда:
 CommandName arg1 arg2 ... argN
- Базовый тип данных — строки
- Контейнеры:
 - список строк,
 - ассоциативный массив
- Функциональная парадигма используется редко
 - Функции высшего порядка и абстракции функции
- Модель событийно-ориентированного программирования
 - Tcl/Tk — библиотека для задания GUI

- Примеры:

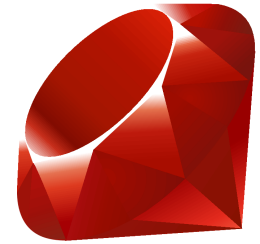
```
proc gcd { a b } {  
    while { $b > 0 } {  
        set t $b  
        set b [expr $a % $b]  
        set a $t  
    }  
    return $a  
}  
puts [gcd 42 56]  
  
set number 5  
set string "Some string with  
    number $number"  
  
set string_list {e1 e2 e3}  
set map_var(one) 1  
set map_var(two) 2  
puts $map_var(two) # «2»
```


Tcl. Применение



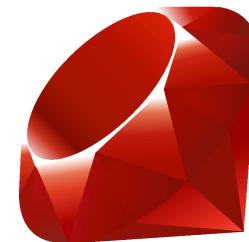
- Применение Tcl:
 - Быстрое прототипирование
 - Создание графических интерфейсов (Tcl/Tk)
 - Встроенный скриптовый язык
 - Тестирование
 - Написание хранимых процедур PostgreSQL (PL/Tcl)
 - Иногда создание CGI-скриптов
- Tcl используют:
 - BMW — пользовательские интерфейсы и интегрирующая среда системы управления производственными процессами реального времени на заводах
 - AOL — встроенный язык программирования в веб-сервере AOLserver
 - Cisco, HP, Motorola — системы тестирования

Ruby. Общая характеристика



- **Ruby** — высокоуровневый язык общего назначения
- Первоначально разработал японец Юкиhiro Мацумото в 1995 г.
- Акцент на
 - понятность кода,
 - высокую скорость разработки,
 - простоту разработки,
 - автоматизацию решения рутинных задач
- Принцип «наименьшей неожиданности»
- Лаконичный синтаксис
- Динамическая типизация
- Объектно ориентированный, с поддержкой процедурной и функциональной парадигм
- Рефлексивный
- Независимая от ОС реализация многопоточности
- Автоматическое управление памятью. Сборщик мусора
- Интерактивный режим выполнения
- Интерпретируемый. Начиная с версии 1.9 — компилируемый в байт-код и выполняемый на виртуальной машине

Ruby. Синтаксис



- Полностью объектно-ориентированный язык:

- все данные — объекты,
- все функции — методы

- Поддержка разделения на модули

- Переменные — имена ссылок на экземпляры классов

- Базовые коллекции:

- массивы,
- хеши,
- другие типы

- Большое количество вариантов коллекций реализовано в стандартных библиотеках

- Примеры:

```
def gcd(a, b)
  while b > 0
    a, b = b, a % b
  end
  a
end

puts gcd(42, 56)
```

```
array_var = [1, 2, 'three']
```

```
puts array_var[1] # "2"
```

```
hash_var = {'one' => 1, 'two' => 2}
```

```
puts hash_var['one'] # "1"
```

```
# Range от 0 до 10
```

```
(0..10).
```

```
# новый Range из квадратов
```

```
collect{|v| v ** 2 }.
```

```
# новый Range из некоторых элементов старого
```

```
select{ rand(2) == 1 }.
```

```
# для каждого элемента печать индекса и самого элемента
```

```
each_with_index{|v,i| printf "%2d\t%2d\n", i, v }
```

```
# Вывод:
```

```
# 0    0
```

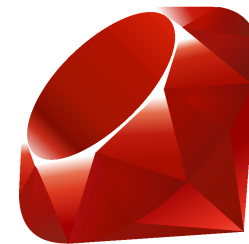
```
# 1    4
```

```
# 2    9
```

```
# 3   16
```

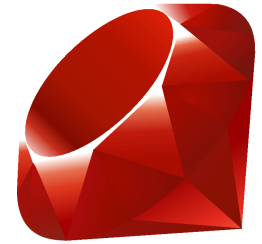
```
# 4   64
```

Ruby. Особенности



- Объектно-ориентированный
 - Инкапсуляция
 - Доступ к данным через методы
 - Public, private, protected члены
 - Вызов метода — передача сообщения
 - Абстракция — утиная типизация
 - Наследование
 - Механизм «примесей» вместо множественного наследования
 - Полиморфизм
 - Возможность динамического изменения содержимого класса, экземпляра класса
- Встроенная в синтаксис поддержка регулярных выражений
- Блоки — именованные замыкания. Итерирование
- Константы
- Механизм обработки исключений
- Встроенная в синтаксис поддержка парадигм функционального программирования
 - Функции высшего порядка
 - Лямбда-функции
 - Continuation
- Поддержка многопоточности
- Метaprogramмирование
 - Метаклассы
- «Синтаксический сахар»
- Существенный недостаток: медленный

Ruby. Применение



- Применение Ruby:
 - Иногда используется как основной язык программирования
 - Веб-программирование (Ruby on Rails)
 - Встроенный скриптовый язык
- Ruby используют:
 - Как встроенный скриптовый язык:
 - Google SketchUp, Inkscape, Xchat, RPG Maker, Amarok
 - NASA — скриптование симуляций
 - Motorola — скриптование эмуляторов, обработка данных

Lua. Общая характеристика



- **Lua** — высокоуровневый язык общего назначения
- Разработан подразделением Tecgraf Католического университета Рио-де-Жанейро в Бразилии в 1993 г.
- Рефлексивный
- Мультипарадигмальный:
 - императивный,
 - функциональный,
 - объектно-ориентированный
- Предоставляет гибкие возможности для расширения языка
- Динамическая типизация
- Компилируется в байт-код, затем выполняется на виртуальной машине

Lua. Синтаксис



- Ограниченный набор базовых типов данных
- Единственный вид контейнера — таблица — ассоциативная структура данных отображающая ключ в значение
- Таблицы используются для реализации
 - структур,
 - областей видимости,
 - массивов,
 - парадигмы объектно-ориентированного программирования
- Примеры:

```
point = { x = 10, y = 20 }  
print(point["x"])      -- "10"  
print(point.x)         -- "10"
```

```
array = { 1, 2, "three" }  
print(array[1])         -- "1"
```

```
function Point(x, y)  
    return { x = x, y = y }  
end  
array = { Point(10, 20), Point(30, 40), Point(50, 60) }  
print(array[2].y)       -- "40"
```

```
function gcd(a, b)  
    while b > 0 do  
        a,b = b, a % b  
    end  
    return a  
end
```

```
print(gcd(42, 56))
```

Lua. Особенности



- Метатаблицы
- Поддерживает парадигмы функционального программирования
 - Функции высшего порядка
 - Замыкания
- Использует небольшие объёмы памяти

Lua. Применение



- Применение Lua:
 - Встроенный скриптовый язык (в основном в видео играх)
 - Язык скриптования для встраиваемых систем
- Lua используют:
 - Игры:
 - Baldur's Gate, Crysis, Far Cry, Ragnarok Online, World of Warcraft
 - Apache HTTP Server — можно использовать для обработки запросов
 - Cisco — реализация Dynamic Access Policies в ASA
 - Awesome — для задания конфигурации
 - Logitech — для скриптования биндингов некоторых клавиатур
 - ASRC Aerospace — сбор и обработка информации для отслеживания утечек опасных газов в ходе запуска космических кораблей

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!