Scripting Languages

Shane Blackett

Bioengineering Institute
University of Auckland
24 November 2006

- PERL
- Python
- Matlab
- TCL/TK
- Ruby
- VB
- Javascript
- PHP

PERL

- Used in CMISS
- Intended for text
- mod_perl in web servers
- resources in Institute
- Extensive libraries (although no longer favourite)
- many ways to do things “does what you want” hard to understand sometimes
- use strict;

Python

- Object Oriented
- Extensive libraries
- Plone/ZOPE
- Well defined behaviour
- Indentation delimits code blocks
- Some institute experience

Matlab

- Numerical arrays (although Perl and Python have packages for arbitrary precision maths, vectors and so on).
- Costs money (although Octave runs .m files)
- C like functions for strings

TCL/TK

- Designed to add control and scripting to existing programs
- if is a function
- TK is badly supported in win32

Ruby

- Fad web scripting language
- Everything is an Object
- Ruby on Rails

VB

- Windows only (except Mono)
- BASIC syntax without line numbers

Javascript

•Browsers Mozilla/Firefox/XUL and IE but different

•Some bad Object Oriented behaviour

•No typechecking except for new versions

•ECMAScript

PHP

- Apache web server
- Inbuilt functions for form handling etc.

Features of languages

- Scripting
- Access to Operating System
- Regular Expressions (nedit, emacs)
- Arrays and Hashes
- Threads
- “Slurping of text”

Access to operating system

- open INPUT_FILE, “bob.txt”;
  while (defined ($line = <INPUT_FILE>))
  {
  }
  close INPUT_FILE;
- open PROGRAM_OUTPUT, “my_program|”;
- opendir, mkdir, chmod, stat
- bind, accept, connect
- fork, pipe, wait

Regular Expressions

- Allow you to match pieces of text
-
```
if ($variable =~ m/Node:\s+([\+\-\d]+)/)
{
  print “Node number $1\n”;
}
```
- Or substitute one string for another
-
```
$variable =~
   s/Node:\s+([\+\-\d]+)/Point: $1/;
```
- Hard to debug: build them up slowly

Hashes

- Indexed with a key
- $myhash{“fred”} = 10;
  $myhash{“bob”} = 6;
  $myhash{“ryan”} = 50;
- print join “ “, keys %myhash . “\n”;
- if (exists $myhash{“ryan”})

```
{
  print “ryan is $myhash{“ryan”}\n”;
}
```

Slurping of text

```
•print <<EOBLOCK;
This is my text
      I can lay it out however I want *
  I can include variables $x $y $z
 so it is great for writing node files
just put this in a loop and set \$x etc.
EOBLOCK
```

Comparison websites

- http://people.mandriva.com/~prigaux//language-study/syntax-across-languages.html
- http://www.99-bottles-of-beer.net/
- http://merd.sourceforge.net/pixel/language-study/scripting-language/

Python Bottles of Beer

```
#!/usr/bin/env python
# -*- coding: iso-8859-1 -*-
"""
99 Bottles of Beer (by Gerold Penz)
Python can be simple, too :-)
"""

for quant in range(99, 0, -1):
   if quant > 1:
      print quant, "bottles of beer on the wall,", quant, "bottles of beer."
      if quant > 2:
         suffix = str(quant - 1) + " bottles of beer on the wall."
      else:
         suffix = "1 bottle of beer on the wall."
   elif quant == 1:
      print "1 bottle of beer on the wall, 1 bottle of beer."
      suffix = "no more beer on the wall!"
   print "Take one down, pass it around,", suffix
   print "--"
```

PERL Bottles of Beer

```
#!/usr/bin/perl
# Jim Menard     jimm@{bbn,io}.com     (617) 873-4326
http://www.io.com/~jimm/
$nBottles = $ARGV[0];
$nBottles = 100 if $nBottles eq '' || $nBottles < 0;

foreach (reverse(1 .. $nBottles)) {
    $s = ($_ == 1) ? "" : "s";
    $oneLessS = ($_ == 2) ? "" : "s";
    print "\n$_ bottle$s of beer on the wall,\n";
    print "$_ bottle$s of beer,\n";
    print "Take one down, pass it around,\n";
    print $_ - 1, " bottle$oneLessS of beer on the wall\n";
}
print "\n*burp*\n";
```

Matlab Bottles of Beer

a href=http://www.mathworks.com>Click</a> for more information.

```
% MATLAB verion of 99 Bottles of beer
% by Bill Becker

function beer(n);
if nargin<1, n=99; end
for i=n:-1:1,
  disp([int2str(i) ' Bottles of beer on the wall,'])
  disp([int2str(i) ' Bottles of beer,'])
  disp('Take one down and pass it around,')
  if i>1, disp([int2str(i-1) ' Bottles of beer on the wall.']),end
  end
disp('No more bottles of beer on the wall!')
```

Matlab Bottles of Beer

a href=http://www.mathworks.com>Click</a> for more information.

```
% MATLAB verion of 99 Bottles of beer
% by Bill Becker

function beer(n);
if nargin<1, n=99; end
for i=n:-1:1,
  disp([int2str(i) ' Bottles of beer on the wall,'])
  disp([int2str(i) ' Bottles of beer,'])
  disp('Take one down and pass it around,')
  if i>1, disp([int2str(i-1) ' Bottles of beer on the wall.']),end
  end
disp('No more bottles of beer on the wall!')
```

Ruby Bottles of Beer

```
# There's more than one 'nice' way to do it ;-)
# www.ruby-lang.org

puts; puts "   It's beer song time!"; puts

def bottles(n)
  n == 1 ? "#{n} bottle" : "#{n} bottles"
end

@count = 99

@count.downto(1)  {
puts <<BEERSONG
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   #{bottles(@count)} of beer on the wall
   #{bottles(@count)} of beer
   Take one down, pass it around
   #{bottles(@count -= 1)} of beer on the wall
BEERSONG
}

puts "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
puts; puts "   No more beer on the wall :-("
```

VB Bottles of Beer

```
Dim n As Integer
Dim s As String

Width = 6000
Height = Screen.Height * 2 / 3
Top = (Screen.Height - Height) / 2
Left = (Screen.Width - Width) / 2
Caption = "99 Bottles of Beer"
List1.Top = 0
List1.Left = 0
List1.Width = Form1.ScaleWidth
List1.Height = Form1.ScaleHeight

List1.AddItem s & "99 bottles of Beer on the wall,"
List1.AddItem s & "99 bottles of Beeeer..."
List1.AddItem "You take one down, pass it around..."
For n = 98 To 1 Step -1
   s = IIf(n = 1, n & " final bottle", n & " bottles")
   List1.AddItem s & " of Beer on the wall."
   List1.AddItem ""
   List1.AddItem s & " of Beer on the wall,"
   List1.AddItem s & " of Beeeer..."
   List1.AddItem "You take one down, pass it around..."
Next n
List1.AddItem "No more bottles of Beer on the wall."
```

PERL

- Remove carriage returns

```
perl -pi’.orig’ -e ’s/\r//g’ myfile
```

- in-place edit of *.c files changing all foo to bar

```
perl -p -i.bak -e 's/foo/bar/g' *.c
```

- rename.pl s/(\w+).c/$1.c2/ *.c