
Perl & TCL

Vijay Subramanian,
subrav@rpi.edu

Modified from :

http://www.telecom.tuc.gr/nlpcourse/resources/tutorial_material/perl06/01_perl_basics_06.ppt

<http://pages.cpsc.ucalgary.ca/~gongm/CPSC441/NS-2.ppt>

Perl

What is Perl?

- **Practical Extraction and Report Language**
- **Interpreted Language**
 - Optimized for String Manipulation and File I/O
 - Full support for Regular Expressions
- **Running Perl under UNIX**
 - Put the following in the first line of your script

```
#!/usr/bin/perl
```
 - Run the script

```
% perl script_name
```

Basic Syntax

- Statements end with semicolon ‘;’
 - Comments start with ‘#’
 - Only single line comments
 - Variables
 - You don’t have to declare a variable before you access it
 - You don't have to declare a variable's type
-

Sample Usage

```
#!/usr/bin/perl -w
```

```
print 5+2;
```

```
print 5 ** 2; # == 25
```

```
print 5 % 2; # 5 mod 2, or one
```

Scalars and Identifiers

■ Identifiers

- A variable name
- Case sensitive

■ Scalar

- A single value (string or numerical)
- Accessed by prefixing an identifier with '\$'
- Assignment with '='

```
$scalar = expression
```



Strings

■ Quoting Strings

- With ' (apostrophe)
 - Everything is interpreted literally
 - With " (double quotes)
 - Variables get expanded
 - With ` (backtick)
 - The text is executed as a separate process, and the output of the command is returned as the value of the string
-

TCL

Tcl Basics

- Basic syntax
 - `command arg1 arg2 arg3 ...`
 - `command` is either the name of a built-in command or a Tcl procedure
- Three basic steps
 - Argument grouping
 - variable substitution, command substitution
 - Command invocation

Examples

- `set var 5`
 - `=> 5`
 - `set b $var`
 - `=> 5`
 - *The set command is used to assign a value to a variable*
 - **Variable substitution**
 - `i`: the character 'i'.
 - `$i`: the variable `i`.
 - **set** `v1 6`
 - **set** `v2 $v1` (variable substitution)
-

Command Substitution

■ Command Substitution

- ❑ **set** value [**expr** \$v1+ \$v2]
- ❑ **=>** set value 12
- ❑ Rewrite the outer command by using the result of the nested command

■ Operation substitution

- ❑ **set i 5 + 6**
wrong number of arguments
- ❑ **set i {5 + 6}**
5 + 6
- ❑ **set i [5 + 6]**
invalid command
- ❑ **set i [expr 5 + 6]**
11

■ Math Expressions

- ❑ **expr 8 / 2**
 - => 4
- ❑ **set len [expr [string length foobar] + 7]**
 - =>13

Grouping

- **Grouping: group words (argument) together into one argument**
 - **Grouping with curly braces**
 - Curly braces prevent substitution in the group
 - **Grouping with double quotes**
 - Double quotes allow substitution to occur in the group
 - **Example:**
 - **set** a hello => hello
 - **puts** stdout " The length of \$a is [string length \$a] ."
 - => The length of hello is 5
 - **puts** stdout { The length of \$a is [string length \$a] . }
 - =>The length of \$a is [string length \$a]
 - **Control Structures**
 - **if** {condition} **then** {.....}
 - **for** {set i 0} {\$i < 10} {incr i 2} {.....}
 - **Procedures**
 - **proc** proc_name {arg1 arg2...} {
-

Example

```
set a 43
set b 27
proc test { a b } {
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}
test 43 27
```

Results

$k < 5$, pow = 1.0

$k < 5$, pow = 1120.0

$k < 5$, pow = 1254400.0

$k < 5$, pow = 1404928000.0

$k < 5$, pow = 1573519360000.0

$k \geq 5$, mod = 0

$k \geq 5$, mod = 4

$k \geq 5$, mod = 0

$k \geq 5$, mod = 0

$k \geq 5$, mod = 4

OTcl Basics

- Creating a class
 - **Class** class_name
 - **Class** class_name –superclass Base_class
 - Defining instance procedures
 - class_name **instproc** proc_name {args} {.....}
 - Defining instance variables
 - **\$self instvar** variable_name (inside a class method)
-

OTcl Basics

- Creating an instance
 - **set** new_inst [**new** class_name]
 - Calling an instance procedure
 - \$new_inst proc_name {args}
 - Using an instance value
 - \$new_inst **set** v1 10
 - set v2 [\$new_inst **set** v1]
-

Examples

```
Class Mom
```

```
Mom instproc greet {} {  
  $self instvar age_  
  puts "$age_ years old  
  mom: How are you  
  doing?"  
}
```

```
Class Kid -superclass Mom
```

```
Kid instproc greet {} {  
  $self instvar age_  
  puts "$age_ years old  
  kid: What's up, dude?"  
}
```

```
set mom [new Mom]
```

```
$mom set age_ 45
```

```
set kid [new Kid]
```

```
$kid set age_ 15
```

```
$mom greet
```

```
$kid greet
```

```
45 years old mom: How  
are you doing?
```

```
15 years old kid:  
What's up, dude?
```