Basic Perl Scripting

March 9, 2005
Perl=Practical Extraction and Report Language
➢not shell programming
➢use version 5.6

Simple Perl script test.pl
#!/usr/local/bin/perl
print “This is a test \n”

Option 1:
➢chmod +x test.pl
➢test.pl

Option 2:
➢perl test.pl
➢make sure /usr/local/bin/perl is in your path
Perl Variables

Simple variables in Perl can have two types of values: **integers** and **strings**
➢ There are also object variables (maybe see this later)

**Integers**: 1, 2, -10

**Strings**: sequences of characters, quoted either as ' ' or “ .... “
➢ a string in between ' ' has value exactly the sequence of characters in between quotes
➢ “ “ some substitutions occurs

```perl
$i=10;
$s1=' winter for the last $i months ';
$s2=" winter for the last $i months ";
print $i;
print $s1;
print $s2;
```

Result:
10
winter for the last $i months
winter for the last 10 months

```perl
$s3="winter for the last \n $i months"
winter for the last 
10 months
```

\n stands for “new line”
Perl Variables

Important to notice:
➢ Unlike shell scripting, you use $var on the left side of an assignment
   \$i=10
➢ Like in shell scripting, you do not need to make explicit the type of the variable
   \$i=10 # understood as an integer
   \$s="10" # treated as a string
➢ Everything in a Perl script is a statement, and statements must end in semicolon
   \$i=10;
   \$s1=' winter for the last \$i months ';
   \$s2=" winter for the last \$i months ";
   print \$i;
   print \$s1;
   print \$s2;

To echo values on the terminal display, use a print statement: print expr, ...., expr;
print 'winter ', " for the last \$i months, \n", "unfortunately"
winter for the last 10 months,
unfortunately
Perl Variables

Perl automatically converts a string to an integer or the other way around, depending on the context:

```perl
$a = "10"
print " a is $a 
"
$a1 = $a + 20
print "a1 is $a1 
"
$a2 = $a . " months"
print "a2 is $a2 
"
$a3 = $a . $a1
print "a3 is $a3 
"
$a4 = $a3 - 1
print "a4 is $a4"
```

+ only makes sense as an integer operand
.

(Concatenation) only makes sense for strings

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>integer</td>
</tr>
<tr>
<td>a1</td>
<td>integer</td>
</tr>
<tr>
<td>a2</td>
<td>string</td>
</tr>
<tr>
<td>a3</td>
<td>string</td>
</tr>
<tr>
<td>a4</td>
<td>integer</td>
</tr>
</tbody>
</table>
Perl Operators

Arithmetic operators:      +, -, *, /, %, ** (exponent)    integers
                          unary +, -

Assignment operators:      =, +=, -=, *=, /=, %=, **=    integers
                          .=                      strings

Standard comparisons for integers:  <, >, <=, >= , ==, !=
String comparison:  eq, ne, lt, le, gt, ge  (alphabetical order)

✓  "10"==10           # automatic conversion of string "10" to integer 10
✓  " 10  "==10        # automatic conversion of string " 10  " to int 10
✗  " 10  " eq "10"    # fails: first string has extra spaces
✓  " 10  " eq " ."10"."

Logical operators:  && (and), || (or), ! (not)
✓  ("abc" lt "cde") && ("abc" lt "Abc")
Conditionals

```plaintext
if (comparison) {
    statement;
    statement;
    ...
}

$i=1;
# prints in order numbers from 1 to 10, on separate lines
if ($i <= 10) {
    print "$i\n"; $i+=1;
}

$i="1";
until ( $s eq "10000" ) {
    print "$s\n"; $s=$s."0"
}
```

# prints in order numbers from 1 to 10, on separate lines
Loops

while (comparison) {
    statement;
    statement;
    ...
}

$i = 1;
for $i (2, 4, 6) {
    print "$i\n";
}
$i = i + 1;
while ($i <= 10) {
    print "$i\n";
    $i = $i + 1;
}

for var (val, ..., val) {
    statement;
    statement;
    ...
}

for (setup; cond; inc) {
    statement;
    statement;
    ...
}

for ($i = 1; $i <= 10; $i += 1) {
    print "$i\n";
}

for ($i = 1; $i < 10; $i += 1) {
    print "$i\n";
}
Files

Open a file `myin.txt` for reading

```perl
open (inh, "<myin.txt");

inh is a file handler (think of it as a number the system assigns to the opened file)
```

```perl
open (inh;"<myin.txt");
while ($line=<inh>) {
    print "$line";
}
close (inh);
```

#reads the input file myin.txt line by line
# displays each line on standard output
Open a file `myout.txt` for writing  
- if the file does not exist, it creates it  
- if the file exists, it overwrites it  
- open a file and append information to it  

```perl
open (outh, ">>myout.txt");

open (inh,"<myin.txt");
open (outh,">>myout.txt");
while ($line=<inh>) {
    print outh "$line"
    # reads the input file myin.txt line by line  
    # appends each line to the output file
}
close (inh);
close (outh);
```
Files

Dealing with errors in opening files:

```c
if (! open (inh,"<myin.txt")) {
    print "Error opening myin.txt!\n";
    exit (1);
} else {
    if (! open (outh,">>myout.txt")) {
        print "Error opening myout.txt!\n";
        exit (1);
    } else {
        while ($line=<inh>) {
            print outh "$line";
        }
        close (outh);
    }
}
```

```c
close (inh);
```