Perl

UVic SEng 265

Daniel M. German

Department of Computer Science

University of Victoria

dmgerman@uvic.ca

The Power of Perl

- Change all `gopher` to `World Wide Web` in a single command

```
perl  -e 's/gopher/World Wide Web/gi'  -p  -i.bak  *.html
```

- Perl can be used as a command
- Or like an interpreter

dmgerman@uvic.ca

Perl

```
#!/public/bin/perl -w
# Suppose this program was invoked with the command
#     go2www ax.html  big.basket.html  candle.html
# Then builtin list @ARGV would contain three elments
# (’ax.html’, ’big.basket.html’, ’candle.html’)
# These could be accessed as $ARGV[0] $ARGV[1] $ARGV[2]

$original=’gopher’;
$replacement="World Wide Web";
$nChanges = 0;

# The input record separator is defined by Perl global
# variable $/.  It can be anything, including multiple
# characters.  Normally it is "\n", newline.  Here, we
# say there is no record separator, so the whole file
# is read as one long record, newlines included.
undef $/;


foreach $file (@ARGV) {
    if (! open(INPUT,"<$file") ) {
        print STDERR "Can’t open input file $bakfile\n";
        next;
    }

   # Read input file as one long record.
```


dmgerman@uvic.ca

```
    $data=<INPUT>;
    close INPUT;

    if ($data =~ s/$original/$replacement/gi) {
        $bakfile = "$file.bak";
        # Abort if can't backup original or output.
        if (! rename($file,$bakfile)) {
            die "Can't rename $file $!";
        }
        if (! open(OUTPUT,">$file") ) {
            die "Can't open output file $file\n";
        }
        print OUTPUT $data;
        close OUTPUT;
        print STDERR "$file changed\n";
        $nChanges++;
   } else {
       print STDERR "$file not changed\n";
   }
}
print STDERR "$nChanges files changed.\n";
```


dmgerman@uvic.ca

Perl Data Structures

- Scalars can be numeric or character as determined by context:

```
123  12.4  5E-10  0xff (hex) 0377 (octal)

'What you $see is (almost) what \n you get'
'Don\'t Walk'

"How are you?"
"Substitute values of $x and \\n in \\" quotes."

`date`   `uptime -u`
`du -sk $filespec | sort -n`

$x    $listOfThings[5]     $lookup{'key'}
```


dmgerman@uvic.ca

Quotes

- Similar to bash:
  - Single-quotes ' ' allow no substitution except for \ and \'.
  - Double-quotes " " allow substitution of variables like $x and control codes like \n (newline).
  - Backticks ` ` also allow substitution, then try to execute the result as a system command, returning as the final value whatever the system command outputs.

dmgerman@uvic.ca

Lists (Arrays)

- Arrays of scalars (also called lists) are sequentially-arranged scalars

```
('Sunday', 'Monday', 'Tuesday', 'Wednesday',
   'Thursday', 'Friday', 'Saturday')

(13,14,15,16,17,18,19)   equivalent to (13..19)

(13,14,15,16,17,18,19)[2..4]  equivalent to (15,16,17)

@wholeList
```


dmgerman@uvic.ca

Associative Arrays

- Also known as hashes

```
$daysInMonth{'January'} = 31;
$enrolled{'UVic'} = 311;
$enrolled{'Waterloo} = 128;

$studentName{654321} = 'Joe Doe';

$score{$studentNo,$examNo} = 89;

%wholeAssociativeArray;
```


dmgerman@uvic.ca

Variables

- Scalars start with $, even when refererring to one element of an array `$wholeList[3]`
- An list is referred with
- An assoc. array is referred with %

```
@days = (31,28,31,30,31,30,31,31,30,31,30,31);
                # A list with 12 elements.
$#days          # Last index of @days; 11 for above list
scalar(@days)   # Number of elements in @days (12)
$#days = 7;     # resets length of list @days to 8 elements
@days           # ($days[0], $days[1],... )
@days[3,4,5]    # = (30,31,30)
```


dmgerman@uvic.ca

Context

- Every operation that you invoke in a Perl script is evaluated in a specific context
- How that operation behaves may depend on the requirements of that context.
- There are two major contexts: scalar and list
- An assignment to a scalar variable, an element of an array or a hash evaluates the right to a scalar context

```
@a = (1, 2, 3,4);
$x          = @a;  # scalar context  $x = 4  (size of @a)
$x[1]       = @a;  # scalar context  $x[1] = 4
$x{"ray"}   = @a;  # scalar context  $x{"ray"} = 4
@x          = @a;  # list context    @x = (1,2,3,4)
%x          = @a;  # list context    %x = (1=>2,3=>4)
```


dmgerman@uvic.ca

Context

- The context can be either integer or string

```
@a = (1, 2, 3,4);
$x  = @a;  # scalar integer context  $x = 4  (size of @a)
$x  = "@a";  # scalar string context $x = "1 2 3 4"
$x  = "$y";  # $y undefined...       $x = "";
$x  = $y + 0;  # $y undefined...     $x = 0;
```


dmgerman@uvic.ca

What is true? Boolean context

- A scalar value is true if it is not the null string "" or the number 0 (or its string equivalent, "0").
- An undefined value (often called undef) is always false because it looks like either "" (empty string) or 0, depending on whether you treat it as a string or a number.
- List values have no Boolean value because list values are never produced in a scalar context (@x results in the size of the array x in an scalar context)

dmgerman@uvic.ca

String Comparison Operators

Operation	Integer	String
Less	<	lt
Less or equal	<=	le
Bigger	>	bt
Bigger or equal	>=	be
Equal	==	eq
Different	!=	ne

```
"a"  ne "b"     # true
"a"  != "b"     # false
```


dmgerman@uvic.ca

Simple Statements

- An expression is a simple statement
- Can be optionally followed by:

```
if EXPR
unless EXPR
while EXPR
until EXPR
foreach LIST
```

- Examples:

```
die "fast" if $youBetrayMe;
$expression++ while $j > 0;
kiss('me') until $iDie;
```


dmgerman@uvic.ca

Compound Statements

- A BLOCK is a sequence of statements
- A BLOCK should be surrounded by braces ({})

```
if (EXPR) BLOCK
if (EXPR) BLOCK else BLOCK
if (EXPR) BLOCK elsif (EXPR) BLOCK ...
if (EXPR) BLOCK elsif (EXPR) BLOCK ... else BLOCK

while (EXPR) BLOCK

for (EXPR; EXPR; EXPR) BLOCK

foreach (LIST) BLOCK
foreach VAR (LIST) BLOCK
```

- Some others, but we don't have time to cover them

dmgerman@uvic.ca

if statement

```
if ((my $color = <STDIN>) eq ’red’) {
    $value = 0xff0000;
} elsif ($color eq ’green’) {
    $value = 0x00ff00;
} elsif ($color eq ’blue’) {
    $value = 0x0000ff;
} else {
    warn "unknown RGB component ‘$color’, using black\n";
    $value = 0x000000;
}
```


dmgerman@uvic.ca

The default variable: $_

- $_ is used as default in many operations
- For example:

```
while (<>) {
    print;
}
foreach ("Hello", "I", "am", "Daniel") {
    if (/Hello/) {
        print;
    }
}
```


dmgerman@uvic.ca

Functions

```
#!/public/bin/perl
foreach (Array()) {
    print Max($_,4) . "\n";
}
sub Max
{
    local($parmA, $parmB) = @_;
    if ($parmA < $parmB) {
        return $parmB;
    } else {
        return $parmA;
    }
}
sub Array
{
    return (1,10,100);
}
```


dmgerman@uvic.ca