Script Programming with Perl

15-123

Systems Skills in C and Unix

What is Perl

- A scripting language
  - Practical extraction and Report Language
  - Developed as a tool for easy text manipulation
- Why Perl
  - Manipulate text — Database programming
  - Files and Processes
- Standard on Unix — man perlintro
- Free download for other platforms

What’s good for Perl?

- Scripting common tasks
  - passwd Files processing
- Tasks that are too heavy for the shell
  - not many tools
- Too complicated (or short lived) for C
  - too many tools

C

|

perl

|

Shell

First Perl Program

```
#! usr/bin/perl  –w
print (“hello world \n”);
# Comment
```

warning

hello.pl

- How does this work?
  - – Load the interpreter
  - – Compile the program with the interpreter
  - – Execute the program
    - perl hello.pl

↳ interpreter

Perl Types

- Variables
  - Names consists of numbers, letters and underscores
  - Names cannot start with a number
- Variable types
  - Scalars
    - Numeric : 10, 450.56
    - Strings
      - 'hello there\n' ← literal
      - "hello there\n"

Perl Data Types

- Scalars
  - strings and numerics
- arrays of scalars
  - ordered lists of scalars indexed by number, starting with 0 and with negative subscripts counting from the end.
- associative arrays of scalars, a.k.a``hashes''.
  - unordered collections of scalar values indexed by their associated string key.

Variables

- $a = 1; $b = 2;
- All C type operations can be applied
  - $c = $a + $b; ++$c; $a +=1;
  - $a ** $b - something new? $\$a^{\$b}$ pow(a,b)
- For strings
  - $s1 . $s2   - concatenation
  - $s1 x $s2 - duplication   "suna" x 3 = "suna suna suna"
- $a = $b ;
  - Makes a copy of $b and assigns to $a

Useful operations

- **substr($s, start, length)**

  Substr("gunawardena", 0, 4) → "guna"

  – substring of $s beginning from **start** position of **length**
- **index(string, substring, position)**

  index(, "dena", 0) → 7

  look for first index of the substring in string starting from position
- **index(string, substring)**

  look for first index of the substring in string starting from the beginning
- **rindex(string, substring)**

  position of substring in string starting from the end of the string
- **length(string) –** returns the length of the string

More operations

$name = "guna";

- **$_ = string; tr/a/z/; # tr is the transliteration operator** $_ = $name;

  replaces all ‘a’ characters of string with a ‘z’ character and assign to $1.

  tr/a/z/;

- **$_ = string; tr/ab/xz/;**

  replaces all ‘a’ characters of string with a ‘x’ character and b with z and assign to $1.

  print $_;

  ↓

  "gunz"

- **$_ = string; s/foo/me/;** Subst

  replaces all strings of “foo” with string “me”

- **chop**

  this removes the last character at the end of a scalar.

  chop ($name)

- **chomp**

  removes a newline character from the end of a string

  Foo, guna, gunand, and_d

- **split splits a string and places in an array**

o @array = split(/:/,$name); # splits the string $name at each : and stores in an array

o **The ASCII value of a character $a is given by ord($a)**

Comparison Operators

Comparison	Numeric	String
Equal	==	Eq
Not Equal	!=	Ne
Greater than	>	Gt
Less than	<	Lt
Greater or equal	>=	Ge
Less or equal	<=	Le

```
$A
$a
```

```
$a = 1;
$b = 2;
if ($a == $b)
```

```
$s1 = "me"
$s2 = "jon"
if ($s1 eq $s2)
```

Operator Precedence and Associativity

Associativity	Operator
left	terms and list operators (leftward)
left	->
nonassoc	++ --
right	**
right	! ~ \ and unary + and -
left	=~ !~
left	* / % x
left	+ - .
left	<< >>
nonassoc	named unary operators (chomp)
nonassoc	< > <= >= lt gt le ge
nonassoc	== != <=> eq ne cmp
left	&
left	| ^
left	&&
left	||
nonassoc	.. ...
right	?:
right	= += -= *= etc.
left	, =>
nonassoc	list operators (rightward)
right	not
left	and
left	or xor

source: perl.com

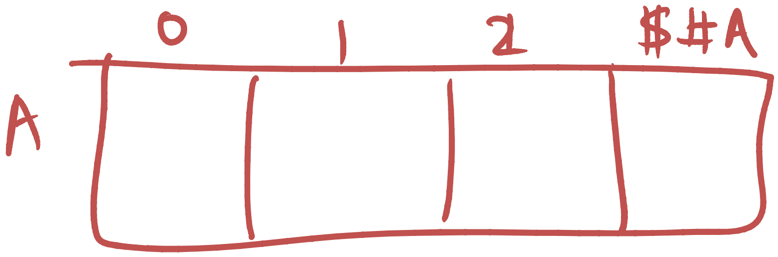
Arrays

$

- @array = (10,12,45);

$#array = 8

- @A = (‘guna’, ‘me’, ‘cmu’, ‘pgh’);
- Length of an array
  - $len = $#A + 1
- Resizing an array
  - $len = desired size



print($A[$#A]) → pgh

$A[-5] = 20;

repetition

A While Loop

```
$x = 1;
while ($x < 10){
print “x is $x\n”;
$x++;
•   }
```

Until loop

```
$x = 1;
until ($x >= 10){
print “x is $x\n”;
$x++;
}
```

repetition

Do-while loop

```
$x = 1;
do{
    print "x is $x\n";
    $x++;
} while ($x < 10);
```

for statement

```
for ($x=1; $x < 10; $x++){
   print “x is $x\n”;
}
```

foreach statement

```
foreach $x (1..9) {
    print "x is $x\n";
}
```

Write a perl program to perform bubble sort – only for fun

```
@A=(3,5,1,2);                              Sort(@A)

for($i=0; $i<$#A+1 ; $i++)
    for ($j=0; $j<$#A+1-i-1 ; $j++)
            if ( $A[$j] > $A[$j+1])

                 Swap
```

Perl IO

```
$size = 10;
open(INFILE, “file.txt”);
$#arr = $size-1; # initialize the size of the array to 10
$i = 0;
foreach $line (<INFILE>) {
  $arr[$i++] = $line;
  if ($i >= $size) {
     $#arr = 2*$#arr + 1; # double the size
     $size = $#arr + 1;
  }
}
```

$2(\$\#A+1)-1$

Perl IO

- open(OUT, “>out.txt”);
- print(OUT, “hello there\n”);
- Better file open
  - open (OUT, “>out.txt”) || die “sorry out.txt could not be opened\n”

Perl and Regex

- Perl programs are perfect for regex matching examples
  - Processing html files
    - Read any html file and create a new one that contains only the outward links
    - Do the previous exercise with links that contain cnn.com only

Perl and regex

```
open(INFILE, "index.html");
foreach $line (<INFILE>) {
  if ($line =~ /guna/ ){
      print $line;
   }
}
close(INFILE);
```

Regex

Regex binding operator

/g+u*n/

Lazy matching and backreference

```
open(IN, “guna.htm”);
while (<IN>){
  if ($_ =~ /mailto:(.*?)"/){
      print $1."\n";
  }
}
```

Global Matching

- How to find all matches on the same line

```
open(IN, “guna.htm”);
while (<IN>){
  if ($_ =~ /mailto:(.*?)"/g){
     print $1."\n";
  }
}
```

Global Matching and Replacing

The statement

$str =~ s/oo/u/;

would convert "Cookbook" into "Cukbook", while the statement

$str =~ s/oo/u/g;

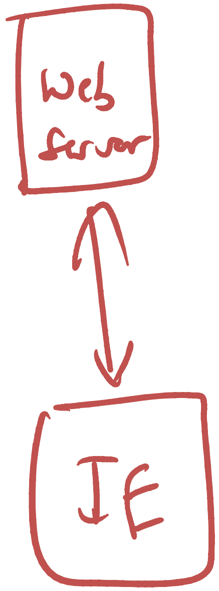
would convert "Cookbook" into "Cukbuk".

CGI Scripts and Perl

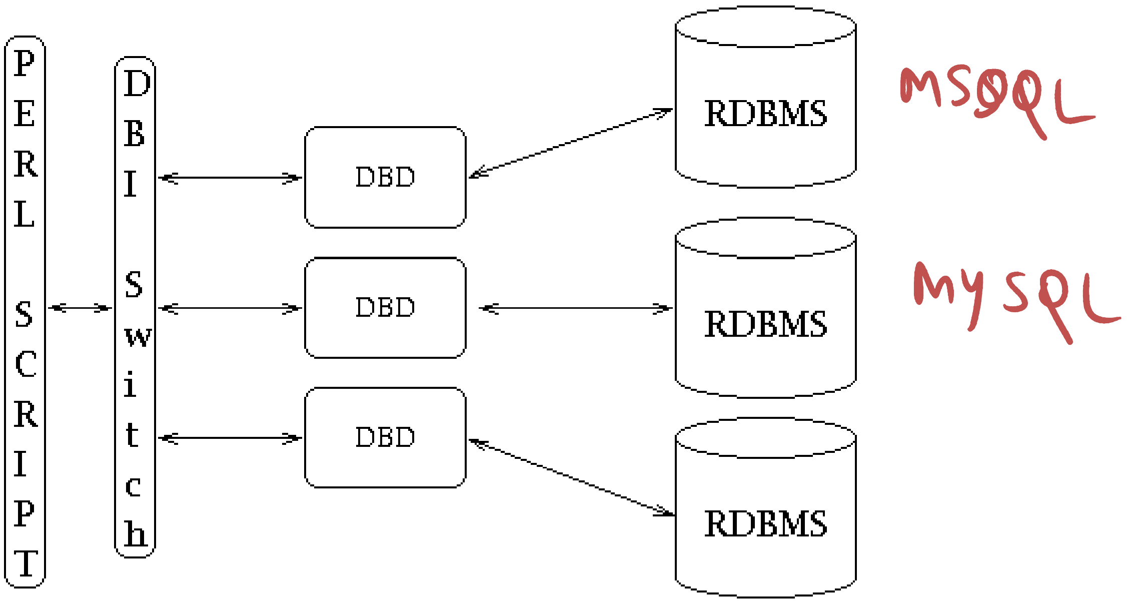
- CGI is an interface for connecting application software with web servers
- CGI scripts can be written in Perl and resides in CGI-bin
- Example: Passwd authentication

```
while (<passwdfile>) {
  ($user, $passwd)= split (/:/, $_);
  ………. if ( $user eq
}
```

guna : 4cs



Perl and Databases



```
<a href="http://www.cs.cmu.edu">
```

More Code Examples