Announcements

- next Monday (October 2) no class
- will be meeting in lab as usual

- first homework assignment will be released online tomorrow...please start early
  - major perl project ...will be using web
Today

- wrap up patterns
- random stuff perl internals
- web based programming

- will be covering object oriented stuff and packages next week

- reading:
  - make sure you understand pattern matching
  - cgi basics
  - object and packages

Small Example

- Many code projects give specific names to version
  - 1.0 dragon
  - 2.0 hawk
  - 3.0 arrow
  - 4.0 camel

- How would you use perl to run through comment and replace all version X.X.X with name information
Code sketch

```perl
%projects = ("1","dragon",
  "2","hawk","3","arrow","4","camel");

if(/^#.*/){
  s/ version\s+(\[0-9\]).*?\s/ The $projects{$1} release
  /g;
}
```

Question ???

- What about those version which we haven’t defined?
Trick: shortcut

- Conditional Operator:
  - COND ? THEN : ELSE

- $a = $b ? $c : $d ;   # ??!!??!

- ?: operator precedence higher than comma

Fix?

```perl
%projects = ("1","dragon",
    "2","hawk","3","arrow","4","camel");

if(/^#.*/){
    s/ version\s+([0-9]).*/$projects{$1}?$projects{$1} release:$&/g;
}
```
Problem

get eye sore looking at it 😊

Fix!

%projects = (^1","dragon",
"2","hawk","3","arrow","4","camel");

if(/^\.#\./){
  s /
    version\s+\([0-9]\).*
  /
    $projects{$1} ?
  The $projects{$1} release
  :
  $&
  /gxe;
}
Careful

- $a = b =~ s/something/else/g;

- Which one is changed?

- Is this what you mean?

Something...any ideas?

1. while s/(\d)(\d\d\d)(?!\d)/$1,$2/;
groups...overriding
- (?#... ) comments
- (?::... ) no capture
- (?imsx-imsx:...) able/dis pattern modifiers
- (?=... ) true if look ahead true
- (?!... ) true if look ahead fails

Group subgroups
$name = "first last";

if($name =~ /((\w+ )\(\w+\))/){
    print "1 is $1\n";
    print "2 is $2\n";
    print "3 is $3\n";
}
Clustering

- Sometimes would like to use parenthesis without capturing
- (?:PATERN)
- Sometimes necessary for operator precedence
- /\^abe|sam|jack/\
- meant to say
- / (?:^\(abe|sam|jack\))/

Quotes (you can quote me)

- Perl has 3 different quote operators
- Can either use the quotes or the function name
- Single quotes
  - " 
  - q{ }
  - Literal meaning, no interop
- Double quotes
  - qq{ }
- Back quotes
  - qx { }
- Word lists
  - qw { }
Some perl

- `qw / /`
  - Will take all tokens between slashes and make "" quotes around things
  - Very useful shortcut when lazy i.e. when you have better things to do 😊

- How do you look up new perl commands ?

Helpful stuff

- `$| = 1`
  - will turn off output buffering great when working with cgi (later today)

- In perl, can call external commands i.e. we can execute command line arguments
  1. Backticks (` `)
  2. System
  3. exec
my Keyword

- Declares the variable lexically scoped
- Only in existence within the current block
- Will be released from memory when we leave the current scope
- Bound from inside out of code blocks

- Rule: Apply maximum limitation on variables

Example

```perl
my $x = 10;

{ 
    my $x = $x;
    $x++; 
    print "here, x is $x \n";
}

print "here, x is $x \n";
```
our Keyword

- Variable which will be global in nature
- Can be created within a block, but will be available anywhere globally

Example

```perl
sub1();
sub2();

sub sub1(){
    our $t;
    $t = 19;
}

sub sub2(){
    our $t;
    print "will print $t\n";
}
```
local Keyword

- Allows you to mix global availability with local temporary values.

- Will take a global variable and use a temp value during current scope
  - Will revert to old value once current scope ends

Example

```perl
use strict;
our $test = "little";

TESTBLOCK: {
    local $test = "temp values";
    print "Test is $test\n";
    sub1();
}

print "We now see $test\n";

sub sub1(){
    print "Now in sub1\n";
    print "we see test as $test\n";
}
```
Slicing

- similar to ranges, can fetch set of values from hash by preceding hash variable with @ sign

```perl
$phonebook;
#do bunch of reads/inserts
@numbers = @phonebook($n1, $n2, $n3);

@phonebook($n1, $n2) = (718,516);
```

What is this exactly?

```perl
$animals = [
    'dog', 'cat',
    'duck', 'cow',
    'pig', 'lizard'
];

$sounds = {
    dog => 'bark',
    cat => 'meow',
    duck => 'quack'
};

@domestic = @{$sounds}{@{$animals}[0,1]};
```
Example of switching warning

```
#beginning of code
use warnings;

#bunch of stuff
{
  no warnings;
  #bunch of other stuff
}
use warnings;
#bunch of other other stuff
```

Something Interesting:

- Can have a perl program with $name @name %name
- All in the same scope
- Perl will never mix them up (that is our job)
How does he do it?

Packages

- Think of a package as an area code for your variables
- Default package is main
- Each package has a symbol table holding its variables
- package FOO;
  - Sets the current symbol table till end of block or next package declaration
  - Can have multiple package declaration
Symbol Table

- This is a data structure which maps variables to information needed by compiler to handle it
- Perl maps variables names to Glob type
- Glob type matches to each variable type
- Each namespace has own symbol table
- Will come back to this later when talking about object creation (will also play with it in the labs)
$package::variable to refer to specific variable
$::variable  #assumes main
$main’something  #old convention
As we say (displaysymbol.pl) main hold global variables
_variables used to be main now anywhere

---

**Little more on ST**

- Symbol tables simple hashes
- All symbol tables linked through main (through parent)
- %main:: has reference to itself
- %main::main::main::main is ok 🤪
- Values are type globs
Short Example..please try it

```perl
sub dispSymbols {
    my($hashRef) = shift;
    my(%symbols);
    my(@symbols);
    %symbols = %{$hashRef};
    @symbols = sort(keys(%symbols));
    foreach (@symbols) {
        printf("%-10.10s| %s\n", $_, $symbols{$_});
    }
}
dispSymbols(%Foo::);

package Foo;
$bar = 2;
sub baz {
    $bar++;
}
```

- switch gears
WWW

- global information space
- URI identify resources available
  - simple representation
  - simple references
  - simple access
- available over the internet
- Client server model
- Document Markup Language

Boring vs. Exciting

- Typical
  - Request is served from a file formatted in html
  - Static file of what we would like to render on a web client.
  - Example:
    - Class syllabus

- What is we could tailor each users web experience to what they want.
  - Design of protocol to handle this
  - dynamic content
CGI

- Common Gateway Interface
  - protocol to allow software to interact with information sources

How does CGI work:

1. HTTP Request
2. Call CGI
3. CGI Responds
4. HTTP Response
Perl + cgi

- Remember:
  - Perl is only a tool here
  - Don’t just memorize, understand
    - Why
    - What
    - How
  - Don’t be afraid to experiment
- STDIN
  - Contents passed to perl script
- STDOUT
  - Will need HTTP headers before printing
- STDERR
  - Depends on server, sometimes just error logs, sometimes error reports on client

%ENV

- This is your best friend in PERL CGI
- Way of getting information from the client
- Create content is way to pass back information to the client
Remember

- Unix permissions
  - user
  - group
  - other
- Need to set permissions:
  - chmod 0755 ????.cgi
    - rw-r-xr-x
- Need to place script in correct place
  - Usually cgi-bin/ directory
- Naming
  - Usually need to end in .cgi

Sample test4.cgi

```perl
#!/usr/local/bin/perl
use strict;

my $time = localtime;
my $remote_id = $ENV{REMOTE_HOST}| $ENV{REMOTE_ADDR};

print "Content-type: text/html\n\n";

print <<END_OF_PRINTING;
This is the time : $time
<P>
and your id is $remote_id
END_OF_PRINTING
```
Some CGI Environmental Variables

- CONTENT_LENGTH
  - Length of data passed to cgi
- CONTENT_TYPE
- QUERY_STRING
- REMOTE_ADDR
  - Ip address of client
- REQUEST_METHOD
- SCRIPT_NAME
- SERVER_PORT
- SERVER_NAME
- SERVER_SOFTWARE
- HTTP_FROM
- HTTP_USER_AGENT
- HTTP_REFERER
- HTTP_ACCEPT
Problem

- How can we print out all the environment variables?

Example

```perl
#!/usr/local/bin/perl
use strict;

my $vars
print "Content-type: text/html\n\n";

foreach $vars (sort keys %ENV){
    print "<P><B>$vars</B><BR>
    print $ENV{$vars};
}
```
HTML

- Hyper Text Markup Language
- Standard by w3: [http://www.w3.org/MarkUp/](http://www.w3.org/MarkUp/)
- Way of standardizing format of documents so that users can share information between different systems seamlessly
- Evolving to XHTML format
HTML

- Hypertext Transfer Protocol
- Language used between web servers and web clients
- http url’s

```
http://www.google.com:80/search?q=what
```

Google.com

- http://www.google.com/search?q=shlomo
Very basics

- Html consists of matching tags
- `<something>` = opening tag
- `</something>` = close tags

- HTML DOC:
  - `<html> <body> .... </body> </html>`

Web pages

- `<title> .... </title>` (before the body section)
- `<H1> .... </H1>` (header titles h1, h2, h3)
- `<P>` paragraphs
- `<BR>` line breaks
- `<b>` ... `</b>` bold
- `<i>` ... `</i>` italicize
- `<u>` ... `</u>` underline
More basics

- `<img src="....." width="X" height="Y">`
- `<a href="www.cnn.com"> something </a>`
- `<a name="Anchor1">` Can be referred to by page.html#Anchor1
- `<hr>` line
- `<hr width=50%>` half line

Lists

- Unordered list
  `<ul> <li> </li> ......<ul>`
- Ordered list
  `<ol> <li> </li> ......<ol>`

- Nested lists
  - Lists themselves can be nested within another
Tables

- `<table>
  <tr>
  <td>Hello</td>
  <td>World</td>
  </tr>
</table>`

comments

<!--
anything you do
-->
More html

- Can get wysiwyg editors
- Word will allow you to save as html
- Can take a look at webpages source code

Browser Issues

- Although HTML should be universal, there are occasional differences between how Microsoft IE renders a webpage and Mozilla firefox
Task…how would you?

1. Create a webpage counter (saying you are visitor x to this page)

2. Now create a graphical counter

---

MD5 Sum

- MD5 – uses a 128 bit hash value
- Designed in 1991
- Known problems with collision attacks
Bottom line

- Still in very wide use
- Allows authentication of files given a file and signature
- Visually authentication against tampering
- What obvious weakness??

Md5 of a file

- Can execute md5sum within perl
- Can use perl defined methods
  - Write yourself
  - Find someone else’s 😊
  - perl libraries....will cover in labs
Using Perl Libraries

[Diagram of CPAN with search results]

Results: 11 of 987 found
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

JavaScript::MD5
Modifies the MD5 digest of a CSS form field
Version: 0.47 MD5-0.47 07 Jul 2004 - Ron Epperson

MD5
Perl interface to the MD5-Message Digest Algorithm
Version: 0.66 MD5-0.66 31 Mar 2003 - Craig Kerr

Digest::Perl::MD5
Perl implementation of Perl-Biased MD5 Algorithm
Version: 0.55 MD5-0.55 17 Dec 2004 - Kyle Atkinson

Digest::MD5
Perl interface to the MD5 Algorithm
Version: 0.55 MD5-0.55 17 Dec 2004 - Kyle Atkinson

Axel::XSP::MD5
Axel XSP utility for MD5 digests
Version: 0.04 MD5-0.04 26 Aug 2001 - Bryce Heaton

Digest::MD5::File
Perl extension for getting MD5 sums for files and links
Version: 0.05 MD5-0.05 26 Aug 2001 - Bryce Heaton

Crypt::MD5::File digests digests
Version: 0.06 MD5-0.06 26 Aug 2001 - Bryce Heaton

Parallel::MD5
Parallel MD5 digests
Version: 0.06 MD5-0.06 26 Aug 2001 - Bryce Heaton

Digest::MD5::File
Perl extension for getting MD5 digests of files
Version: 0.06 MD5-0.06 26 Aug 2001 - Bryce Heaton

Digest::MD5::File
Perl extension for getting MD5 digests of files
Version: 0.06 MD5-0.06 26 Aug 2001 - Bryce Heaton
Digests

- The 128-bit (16-byte) MD5 hashes (also termed message digests) are typically represented as 32-digit hexadecimal numbers.
- Even small change can result in a totally different hash digest
Digests II

- MD5("The quick brown fox jumps over the lazy dog") = 9e107d9d372bb6826bd81d3542a419d6
- MD5("The quick brown fox jumps over the lazy cog") = 1055d3e698d289f2af8663725127bd4b
- MD5("") = d41d8cd98f00b204e9800998ecf8427e

GUI

- There are easy ways to make graphics in perl
- Will not cover in this course
  - But will have enough knowledge to pick this up on your own if you choose
  - Better way: will see later today
Graphics

```perl
#!c:\perl\bin
use Tk;

my $mwin = MainWindow->new;

$mwin->Button(-text => "Hello World!", -
        command => sub{exit})->pack;
MainLoop;
```

- Good to know about
- Might need to one day debug someone else’s code (GASP!)
Computer Security

- System and theory of ensuring the confidentiality, integrity, availability, and control of electronic information and systems.
  - Network
  - Host
  - Data

For host based security

- Want to ensure permission system
  - X should only be allowed to do A, B, and C
- Want to ensure accountability
  - If Y does something not allowed, should be noted
- Want to be able to track
  - If something has been tampered with, how can we locate it
  - Both preventative and reactionary
Homework Project

- Assuming you are a system administrator or just paranoid
- Take chronological snapshots of your system to compare and find changes
  - Many changes by system
  - Many changes by valid user
  - Might locate malicious user/system changes
- Want to search filenames
- Want to organize snapshots of system

Useful programming tips

- use warning
- use strict
- learn to use debugger
- Create debugging statements to help chart progress throughout program...
- be clear about what you are doing...don’t use fancy tricks at beginning
Simple example

- [http://www.cs.columbia.edu/~name/a.pl](http://www.cs.columbia.edu/~name/a.pl)
- User in browser invokes perl script
- Web server calls script
- Perl script runs and print out a html code
- Web browser renders the webpage

Next step

- Not just execute the script want to get some starting information from the user