

CS3157: Advanced Programming

Lecture #3

May 31

Shlomo Hershkop
shlomo@cs.columbia.edu

Outline

- References
- Perl Objects
- More CGI
- Setting up webserver/cgi

Announcements

- We will demo today
Recommended environments
 - Eclipse
 - Perl Plug in EPIC
 - CGI – Webserver

we can do it as soon as I hear snoring 😊

Labs

- Lets talk about one of the problems
- Word statistics:
 - Count the number of unique words in the file.
 - Top 5 occurring words in the file
 - Count the **average** length of words in the file.
- Different approaches

From last time

- We were covering CGI, will pick up soon once we cover references and perl objects

- Any questions from last time ?

References

- A reference is a way of talking about a variable

- Symbolic Reference: a reference which we need to look up what it means in some other place

- Hard Reference: direct value of other object

Creating References

- Backslash (covered last time)
- `$foo = 200;`
- `$ref_foo = \ $foo;`
- `$constref = \312;`
- `$sub_ref = \&some_sub;`

References to Arrays

- `$ref = [0,1,2,3,];`
 - how would you print the 1 ?
- `$ref = [[-1,1],45,78,['s','r']];`
 - how would you print the 's' ?

Hashes

- Remember that when you stick an anonymous list into a hash value (why not key ?)
- How to use it ?

sub references

- `$subref = sub(print "hello\n");`
- `&$subref;`

Binding

- `$$arrayref[0]`
- What does it mean ?
- `$$arrayref[0]` `$$arrayref[0]`

Arrow Operator

- `->` infix operator for referencing
 - arrays
 - hashes
 - subroutines
- `$$arrayref[2]` can be `$arrayref->[2]`
- `&$subref()` can be `$subref->()`

Being Helpful

- Interesting point: if you dereference a value that is not a reference, it will be treated as a global variable (i.e. first use)
- `use strict 'refs';`

Modules

- Idea: take a piece of code we are using a lot, and package it up so anyone can reuse it
- Advantages ??
- Disadvantages ?

Modules

- File with code:
 - something.pm
- Code which wants to use it:
 - use something;
- `really::something::else`
 - would be `really/something/else.pm`

Subroutines in Modules

- something.pm which defines `foo()`;
- either
- `something::foo()`;
- or export it correctly in the module file
 - `@EXPORT = qw(foo)`;

Warning

- Be careful about exporting!!
- Does anyone know the difference between overloading and overriding ??

Bad Example

- say your package defined a sub called isdir
- If you export it what will happen ?

Versions

- can also define a \$VERSION scalar which tells perl what your version is
- use `Function 3.2.1.2;`
- would check the `Function.pm` for version 3.2.1.2 or later

Objects

- Perl can be programmed in an object oriented fashion with objects/classes etc
- Although you end up using references!
- Class is just a package
- Bless!! ties a reference with a class!

Class declaration

- package Person;
- Will define the scope until the end of the current file

methods

- Accessor and modifier methods are defined in the current package
- Use arrow notation to access methods
- Variable => name of method()

```
sub print {
  my ($self) = @_;

  #print info
  print $self->firstName . " ". $self->lastName;
}
```

constructor

```
sub new {
  my $self = {

    _firstName => undef;
    _lastName => undef;

  };
  bless $self, 'Person';
  return $self;
}
```

Instantiating

- `my $shlomo = new Person();`

accessing

```
sub firstName {  
    my ( $self, $firstName) = @_;  
  
    $self->{_firstName} = $firstName if  
    defined ($firstName);  
  
    return $self->{_firstName};  
}
```

Inheritance

- Classes can be used to extend the functionality of a class using a process called inheritance
- @ISA
 - perl keyword showing inheritance

Switch

- Back to CGI
- review: What is CGI ?
- How do we write CGI applications ?

Forms

- One way to get information is to collect data
 - Registration
 - Payment
 - Surveys
- Commands
 - Possible choice combination
 - Actions
- Generally user needs to hit submit for anything to happen

Example

- Google.com

- Load page
- Do nothing...nothing happens
- Type search...nothing happens
 - Hit submit/return trigger action

Other way

- React to user typing (will not be doing this)



2 ways to do it

1. Create a HTML file and display a form, and your script gets input from the form
2. Have your script run
 1. If no information is being passed, print out the html for a form (then end)
 2. Else process the form information in the script

Interacting

- GET
 - HTTP request directly to the cgi script by appending the URL
- POST
 - HTTP request in content of message, i.e it is stdin to your script
- Format of GET (default):
 - Value=key separated by &
 - Space replaced by +
 - URL conversion characters

Input Tag

- Each field is in an input tag
- Type
 - Text
 - Radio button
 - Checkbox
 - Pull down menus
 - etc
- Name
 - Symbolic name (so can recognize it)
- Value
 - Default value, or what the user will end up typing

Encoding

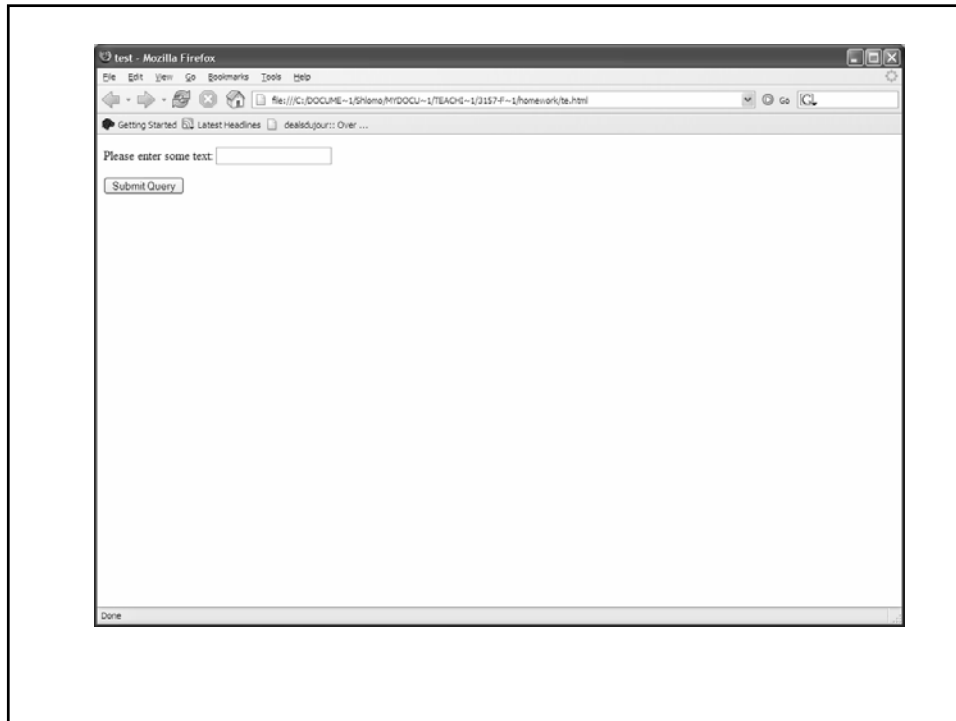
- Spaces are turned to +
- & separates field
- Special characters are turned into %?? (hex)
 - “(“ is %28
 - So “class is great” = “class+is+great”

others

- Submit buttons
 - `<input type="submit">`
- Reset buttons
 - `<input type="reset">`
- Value will change the default name on the button

Putting it all together

```
<form action="cgi/some.cgi" method="GET">  
  <p> Please enter some text:  
  <input type="text" name="string"></p>  
<input type="submit">  
</form>
```

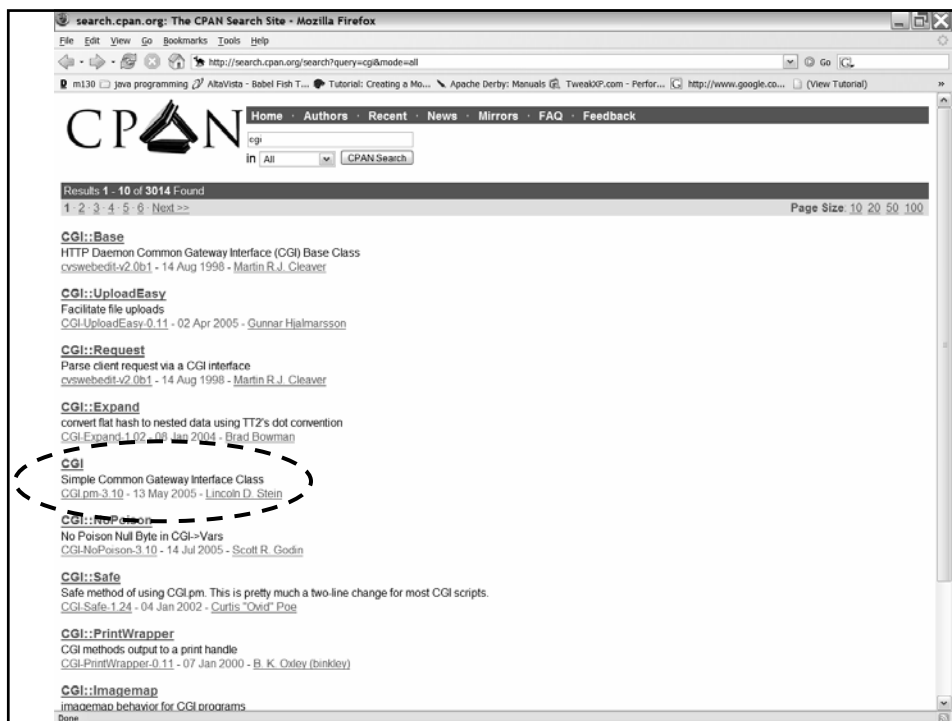


Decoding Form Input

1. `$ENV{QUERY_STRING}`
2. `If($ENV{REQUEST_METHOD} eq POST)`
`{ read $ENV{CONTENT_LENGTH}}`
3. Split pairs around `&`
4. Split keys and values
5. Decode URL
6. Remember key, values

Drawback

- A lot of work
- Pain if we have multiple values associated with one key
- Must be easier way.....
- CGI.pm
 - Included after 5.003_07+



CGI.pm

- Allows you to handle cgi in a standard format
- Can save and load key,value pairs to standard file
- Helps in creating html documents to the server by streamlining certain operations and keeping it in an object oriented design

The bad news

- Can't use it in this class
- Want you to practice doing it the manual way...better for learning and later CGI + C/CPP

Summary: CGI

- Minimum the web server needs to provide to allow an external process to create WebPages.
- Goal: responding to queries and presenting dynamic content via HTTP.

Requirements

- Webserver setup correctly
 - We will practice with Abyss
 - <http://www.aprelium.com/abyssws/download.php>
- Configure the cgi script
 - Will cover this lab.
 - <http://www.aprelium.com/data/doc/2/abyssws-win-doc-html/hosts-configuration.html>
- Basic http/html knowledge

http headers

HTTP/1.1 200 OK

Request / Status Line

Content-type text/html
Content-Length: 300

Header Fields

GET /index.html HTTP/1.1

- GET
- HEAD
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE

Server responses

```
HTTP/1.1 200 OK
Date: Sun, 25 Sep 2005 20:30:12 GMT
Server: Apache/1.3.5 (Unix)
Last-Modified: Wed, 20 May 1998 13:12:11 GMT
ETag: "2345-7227363ed"
Content-Length: 141
Content-Type: text/html
```

```
<HTML>
<HEAD><TITLE>.....
```

CGI Environment

- In perl available through the %ENV global hash
- Changing any of the values will only be seen by your own subprocess
 - Why?
- Some of the variables will be blank
 - Why?

File handling

- We covered basic file handling
- How does this change over the web?

File Locking

```
use Fcntl " flock";
```

```
open FILE, "?????.txt" or die $!;
```

```
#one of these
```

```
flock FILE, LOCK_EX;
```

```
flock FILE, LOCK_SH;
```

```
.....
```

```
flock FILE, LOCK_UN;
```

Side Note: Line Endings

- Carriage return \r
- Line Feed \n
- CRLF
- Unix – LF (\n) CR (\r)

- print “Content-type: text/html\n\n”

- Why not \n\r\n\r ????

Serving web pages

```
#!/usr/local/bin/perl
use strict;
$|=1;

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
```

Serving more than webpages

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

```
print "Content-type: image/jpeg\n\n";
```

```
print "Content-type: image/png\n\n";
```

```
print "Content-type: audio/mp3\n\n";
```

Serving mp3 files

```
open(MP3FILE, "....") || die ....
```

```
my $buffer;
```

```
print "Content-type: audio/mp3\n\n";
```

```
binmode STDOUT;
```

```
while( read(MP3FILE, $buffer, 16384)){
```

```
    print $buffer;
```

```
}
```

Example

- <http://.../cgi-bin/mp3server.cgi/Song.mp3>

Argument passing

- Say you have a cool program which you can hook to the web.....
 - Give a cell phone
 - Give a message
 - Will send the cell phone a message

```
<HTML><HEAD>
<TITLE>Cool</TITLE>
</HEAD>
<BODY>

<form action="cgi-bin/cool.cgi" method="GET">
<p>Enter cell phone to use:
<input type="text" name="cellphone"></p>
<p>Enter Message:
<input type="text" name="message"></p>
<input type="submit">
</form>
</BODY></HTML>
```

```
Use CGI;
my $coolp = '/usr/local/bin/cellmsg';

my $q = new CGI;
my $cell = $q->param("cellphone");
my $msg = $q->param("message");
#error checking here
open PIPE, "$coolp $cell $message |" or die "Can
    not open cellphone program";
print $q->header( "text/plain");
print while <PIPE>
close PIPE;
```

What can go wrong?

- When executing command can in theory pass in the following arguments

Something ; rm -rf *.*

Perl Taint mode

- -T
 - Taints all data references (incoming)
- `#!/usr/bin/perl -wT`
- Flags data to make sure perl doesn't do anything insecure

Tainted?

- STDIN
 - CGI
 - If variables/values are tainted
 - Tainted follows it around with assignments
- ```
Sub is_tainted {
 my $var = shift;
 my $blank = substr($var ,0,0);
 return not eval { eval "1 || $blank" || 1};
}
```



## Why

- Why would you want to keep track of tainted data?

## Getting out of taint

- Match related patterns (\$1,\$2 ..)
- Idea: would check for security problems and then allow it
  
- Reminder: only in taint mode if set

## Other issues

- Remember with each user, your perl script is being instantiated and executed
- In general might want to be able to run alongside yourself (not only in web context).
  - How do we share a variable between instances (to pass information) ?

## Command shell

- A better way of executing command shell arguments to a program is to divide the work
- Create an instance of the program you want to run
- Pass arguments directly to it, instead of using the command shell (where can combine multiple commands)

## fork/exec

```
my $pid = open PIPE, "-|";
die "problem forking $!" unless defined $pid;
```

```
unless($pid) {
 exec COOL, $message or die "cant open
 pipe $!";
}
```

## Some more background

- When you work with CGI, many times you have to work with specific formats and files
- Need to know how it will be handled on client side
- One such common file, is graphics..

# Graphics

- **Formats:**
  - GIF (Graphic Interchange Format)
    - 256 colors
    - LZW compression
    - Animation
    - Transparent bit
  - PNG (Portable Network Graphic)
    - 256 color / 16-bit gray / 48-bit true color
    - NOT LZW
    - Alpha channels
    - Interlacing algorithms

- **JPEG (Joint Photographic Expert Group)**
  - 24-bit color
  - Lossy compression
  - No animation/transparency
- **PDF (Portable Document Format)**
  - Postscript language for document layout

## Image manipulation

- Many packages in perl to work with image data
- GD
  - Lightweight package
  - Port of c graphics library
  - Manipulation routines for PNG

## CGI

- CGI is a common framework
- Perl is not the only player
- We will also be doing CGI + PERL|C|CPP

## Alternatives

- ASP
  - Created by Microsoft for its servers
  - Mix code into html
  - Visual basic/javascript
- PHP
  - Apache webserver
  - Similar to perl
  - Embed code in html

## Alt II

- Coldfusion
  - Webserver interprets std coldfusion call embedded in html, and can add code to run custom functions
  - Windows, and linux
- Java servlets
  - Compiled java classes invoked by web client
  - Code creates documents
- FastCGI
  - Threaded instance of perl continuously running to help cgi perl run faster
- Mod\_perl
  - Apache server perl thread to make perl cgi faster

## Text handling

- One of the exciting developments in the last decade of computer science is data processing/mining/learning
- Many other area in and out of CS need data to be analyzed or presented in some (controlled but arbitrary fashion)

## Handling data

- Using chiseled stone
- By hand (literally copy paste)
- Early mechanics (typewriters)
- Take 3157 ☺

## Outputting text

- Many times will have multiple fields per line
- Arbitrary delimiters:
  - Comma
  - Tabs
  - Pipe |
- Make sure whatever you choose
  - Is either not/can't be present in the data
  - What if it is? How to represent these delimiters ??

## Approach

- Memory vs disk based handling
- Brute force
- Divide and conquer
- Regexp is your friend



## Ahead!

- Because CGI/Internet involves network based thinking, I will illustrate a quick example now.

## Socket

- In order to communicate across computer networks (or between processes on the same computer)
- Need to decide on rules of communication
  - Language of protocol
  - Directional vs bi-directional
  - Throwaway or continuous
  - Life time of communication
  - Overhead
  - Priority
  - Location

## IO:Socket client

see: <http://search.cpan.org/~gbarr/IO-1.2301/IO/Socket/INET.pm>

Use `IO::Socket::INET`;

```
$socket = IO::Socket::INET->new(
 PeerAddr => $remote_host,
 PeerHost => $remote_port,
 Proto => "tcp",
 Type => SOCK_STREAM) or die...
```

```
#writing out
print $socket "hello World";

#notice treatment of handle
$answer = <socket>;

close($socket);
```

## Server version

```
my $server = IO::Socket::INET->new(
 LocalPort=> $portnum,
 LocalAddr => 'localhost',
 Proto => 'tcp',
 Reuse => "1",
 Listen => "10")
 or die "could not start server on port $portnum\

while($client = $server->accept()) {
 #...
}
```

## Next

- Start Lab 2 early
- Make sure you are started on Homework 1