CS3157: Advanced Programming
Lecture #5
Sept 26
Shlomo Hershkop
shlomo@cs.columbia.edu

Outline
• CGI
• CGI security
• CGI Graphics
• Alternative Technologies
• Threading

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
  – Will not talk about it in class.
• Configure the cgi script
  – Will cover this lab.
• Basic http/html knowledge
http headers

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)
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?
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\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

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 $!";
}
### 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

### File Locking

```perl
use Fcntl "flock";
open FILE, "?????.txt" or die $!
#one of these
flock FILE, LOCK_EX;
flock FILE, LOCK_SH;
.....
flock FILE, LOCK_UN;
```
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

Wednesday

• Meet in the clic lab 2-4pm or 4-6pm
  – Please choose a spot in either lab
  – Feel free to bring your own laptop if you want to stay full time
  – Feel free to ask help for anything during lab.
• Make sure you have a cs account.
• Graded lab assignment part of class, will be due Friday afternoon.

Outputting text

• Many times will have multiple fields per line
• Common delimiters:
  – Comma
  – Tabs
  – Pipe |
• Make sure what ever you choose is not in the data
• How to represent these delimiters if they are present??
Socket

- In order to communicate across computer networks (or between processes on the same computer) need to setup a communication address.

- IO::Socket

IO::Socket client

Use IO::Socket::INET;

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

```perl
# writing out
print $socket "hello World";
$answer = <socket>;
```

Server version

```perl
$server = IO::Socket::INET->new( LocalPort=>
    $server_port,
    Type => SOCK_STREAM,
    Reuse => 1,
    Listen = 10) or die....
```

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

- Multi threading
  - Fork processes
  - Process space
- Communication
  - Pipes
  - Sockets