CGI-Perl, GnuPlot, etc - L529
Kind of a "bits n pieces" Tutorial![1]

Arvind Gopu [2]

October 2003

[1]Examples in this are from perldoc.com or Sun Kim's examples
[2]Author Contact Info: http://peart.ucs.indiana.edu/contact.html

Contents

0.1 Introduction 1
0.2 Basic CGI Scripting 2
0.3 Data Processing 5
0.3.1 Naive way (almost specifically Sequence data) 5
0.3.2 Using Multipart (almost specifically Sequence data) . . . 8
0.4 GNUPlot Basics 10
0.4.1 Using GNUPlot inside CGI scripts 15
0.5 LWP 16
0.6 Graph Packages 18

0.1 Introduction

In these web pages I will try to show how you can write simple CGI scripts to do processing over the web. We start off with really basic stuf like HTML element-generation in the next section and then go on to look at methods to upload files, call webpages from within a script, plot data from within a script and so on so forth.

If you are not sure about the server you should work on or which directory you should be creating HTML files in (for L529 class folks) then please look at this page: ...docs/biokdd_init_stuff.html to start with.

Once you are logged into Biokdd as L529guest or yourself, please create your own directory, if it does not exist yet, by doing:

```
General: [L529guest@biokdd home2] $ [ -d /var/www/cgi-bin/some_dir_name_under_L529guest_if_you_
                                    || mkdir /var/www/cgi-bin/some_dir_name_under_L529guest_if
Example: [L529guest@biokdd home2] $ [ -d /var/www/cgi-bin/L529guest/agopu ] || mkdir /var/www/c
```

And then change to your CGI-bin directory by doing:

```
General: [L529guest@biokdd home2]\$ cd /var/www/cgi-bin/dir_you_created_here
Example: [L529guest@biokdd home2]\$ cd /var/www/cgi-bin/L529guest/agopu
         or if you are using your own account...
```

```
[L529guest@biokdd home2]\$ cd /var/www/cgi-bin/agopu
```

Check if you are on the right working directory:

```
[L529guest@biokdd agopu]\$ pwd
/var/www/cgi-bin/L529guest/agopu
```

From this point on, we'll assume you are in this right working directory all through this document unless mentioned otherwise.

0.2 Basic CGI Scripting

In this page I will try to explain the basic elements of a CGI script. Please note that this bit of code uses the Object Oriented way of doing CGI scripting; If you prefer a Functional way, please look at the next subsection - it explains how you can port this code to have a functional structure. The core remains the same, there'll be just a few syntactical changes. If you are not concerned about this, forget it!

To see how this script works, create a file using your favorite editor (pico/vi/emacs/cat/whatever you like!) called "simple_complete.pl" and copy n paste the code shown below into it. I've shown how you can do this using 'cat' here. Haifeng's introduction to CGI (Link off the class home page Lab Section [1]) should get you started nicely on the real basics.

```
[L529guest@biokdd agopu]\$ cat > simple_complete.pl


#!/usr/bin/perl
## This Code from http://www.perldoc.com/perl5.8.0/lib/CGI.html (slightly modified)

use CGI;

$query = new CGI;

print $query->header;
print $query->start_html("L529 - Example CGI.pm Form");
print "<h1> L529 - Example CGI.pm Form</h1>\n";
&print_prompt($query);
&do_work($query);
&print_tail;
print $query->end_html;

sub print_prompt {
    my($query) = @_;

    print $query->start_form;
    print "<em>What's your name?</em><br>";
    print $query->textfield('name');
    print $query->checkbox('Not my real name');

    print "<p><em>Where can you find English Sparrows?</em><br>";
    print $query->checkbox_group(
 -name=>'Sparrow locations',
 -values=>[England,France,Spain,Asia,Hoboken],
 -linebreak=>'yes',
 -defaults=>[England,Asia]);

    print "<p><em>How far can they fly?</em><br>",
    $query->radio_group(
-name=>'how far',
-values=>['10 ft','1 mile','10 miles','real far'],
-default=>'1 mile');

    print "<p><em>What's your favorite color?</em>  ";
    print $query->popup_menu(-name=>'Color',
     -values=>['black','brown','red','yellow'],
     -default=>'red');

    print $query->hidden('Reference','Monty Python and the Holy Grail');

    print "<p><em>What have you got there?</em><br>";
    print $query->scrolling_list(
 -name=>'possessions',
 -values=>['A Coconut','A Grail','An Icon',
   'A Sword','A Ticket'],
 -size=>5,
 -multiple=>'true');

    print "<p><em>Any parting comments?</em><br>";
    print $query->textarea(-name=>'Comments',
   -rows=>10,
   -columns=>50);

    print "<p>",$query->reset;
    print $query->submit('Action','Shout');
    print $query->submit('Action','Scream');
    print $query->endform;
    print "<hr>\n";
}
```

```
sub do_work {
    my($query) = @_;
    my(@values,$key);

    print "<h2>Here are the current settings in this form</h2>";

    foreach $key ($query->param) {
print "<strong>$key</strong> -> ";
@values = $query->param($key);
print join(", ",@values),"<br>\n";
    }
}

sub print_tail {
    print "<hr>";
}
```

Press “Ctrl + d” to exit cat.

Make the perl script executable – change permissions:

```
[L529guest@biokdd agopu]\$ ls -l
total 4
-rw-rw-r--    1 L529guest L529guest     2056 Oct 26 17:44 simple_complete.pl
[L529guest@biokdd agopu]\$ chmod 755 *.pl
```

Alright now let us see how this code works. I will try to go sequentially down the code and explain some of it.

- We start with the usual #! line which specifies where the perl compiler sits.
- The ‘use’ line specifies the package to use. In this case it’s the CGI package.
- As mentioned we’ve shown the object oriented way of scripting CGI (essentially it’s a copy n paste from perldoc!). That is the reason we’ve created a CGI query object. All operations and function calls will be made with respect to this object.
- A general way of calling any CGI function is to use the -¿ operator on the query object. The way you write subroutines is similar the way it’s done on Perl.
- Some of the functions used:
  - header: Writes the HTML header

  - start_html: Start off with the ¡HTML¿¡HEAD¿.... tags. End_html does the closing of these tags.
  - Creation of a textfield, a independent checkbox, a group of checkboxes, radio buttons, a popup menu, hidden variables, a listbox (Scrollable), a textarea and a submit button is shown in that order. If you have doubts about how these work, please refer [2].
  -

- The function do_work() essentially goes through the keys of the param hash table and prints all the key-value pairs. The param hashtable is the one which gets passed from one HTML/CGI script to another.

Try opening the page you just created on your favorite browser. Going to:
http://biokdd.informatics.indiana.edu/cgi-bin/L529guest/agopu/simple_complete.pl
...should get you started. Replace 'agopu' with your user name or whatever you named your directory to be. Remove 'L529guest' if you are using your own account.

0.3 Data Processing

In this section we will look at how you can pass around data between two scripts. Usually this will be sequence data or sequence file names in our case though what we gonna look at, is a general method of doing things.

0.3.1 Naive way (almost specifically Sequence data)

In this subsection we will check out how we can pass data (text pretty much, I guess) from one HTML/CGI script to another using a textfield or textarea. This is quite a naive way of doing it especially as the size of data goes up. Either way we do come across a lot of webpages with such an interface.

Create a HTML file called "/var/www/html/L529guest/agopu/clustalw.html" (as usual replace 'agopu' ....) with the following content. A brief discussion follows the source code.

```
<html>
<title> Clustalw TEST </title></head><body><br>
<nobr>
<h1> Clustalw TEST </h1> <br>

<form action="/cgi-bin/L529guest/agopu/clustalw.pl" method="post">
<input type="submit" value="Do it "><input type="reset" value="Clear">
<br>
Type or cut and paste your sequence in FASTA format in here <br>
<textarea rows=10 cols=50 name="atext">
</textarea>
</form>
```

```
</body></html>
```

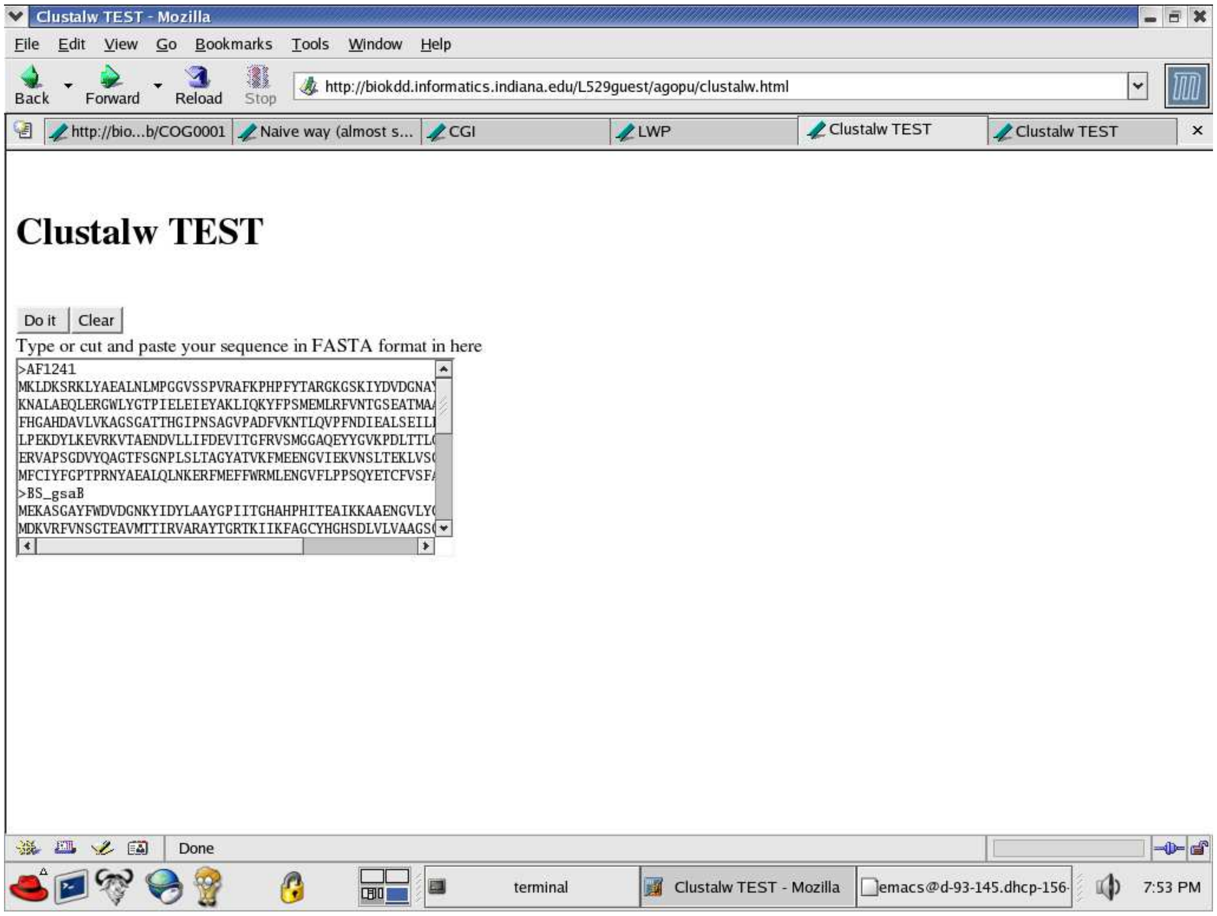
Create a CGI script called "/var/www/cgi-bin/L529guest/agopu/clustalw.pl" (as usual replace 'agopu' ....) with the following content.

```
#!/usr/bin/perl

use CGI qw(:cgi-lib);
ReadParse();

$q = $in{CGI};

my      $CLUSTALW="/usr/local/bio/bin/clustalw";
my      $DIR="/tmp/";
my      $TMPSEQS=$DIR."myseqs".$$;

$|=1;
print "Content-type: text/html\n\n";

print "<pre>";
print "<br> Temp sequence file is $TMPSEQS <br>\n";

open(THISF, ">$TMPSEQS") or die "open $TMPSEQS failed";
print THISF  $q->param(-name=>'atext');
close THISF;

system ("$CLUSTALW $TMPSEQS > $TMPSEQS.out 2>&1");

open(FILE, "$TMPSEQS.aln") or die "open $TMPSEQS.aln failed";
print "<br> Alignment: <br>\n";
while(<FILE>) {
  print "$_";
}
close FILE;
```

Do not forget to change permissions for these files as shown in the previous section:

```
[L529guest@biokdd agopu]\$ chmod 755 /var/www/cgi-bin/L529guest/agopu/*.pl
[L529guest@biokdd agopu]\$ chmod 644 /var/www/html/L529guest/agopu/*.html
```

This is the stuff you can expect to see on your browser if you visit:
http://biokdd.informatics.indiana.edu/L529guest/agopu/clustalw.html



Clustalw TEST

Type or cut and paste your sequence in FASTA format in here

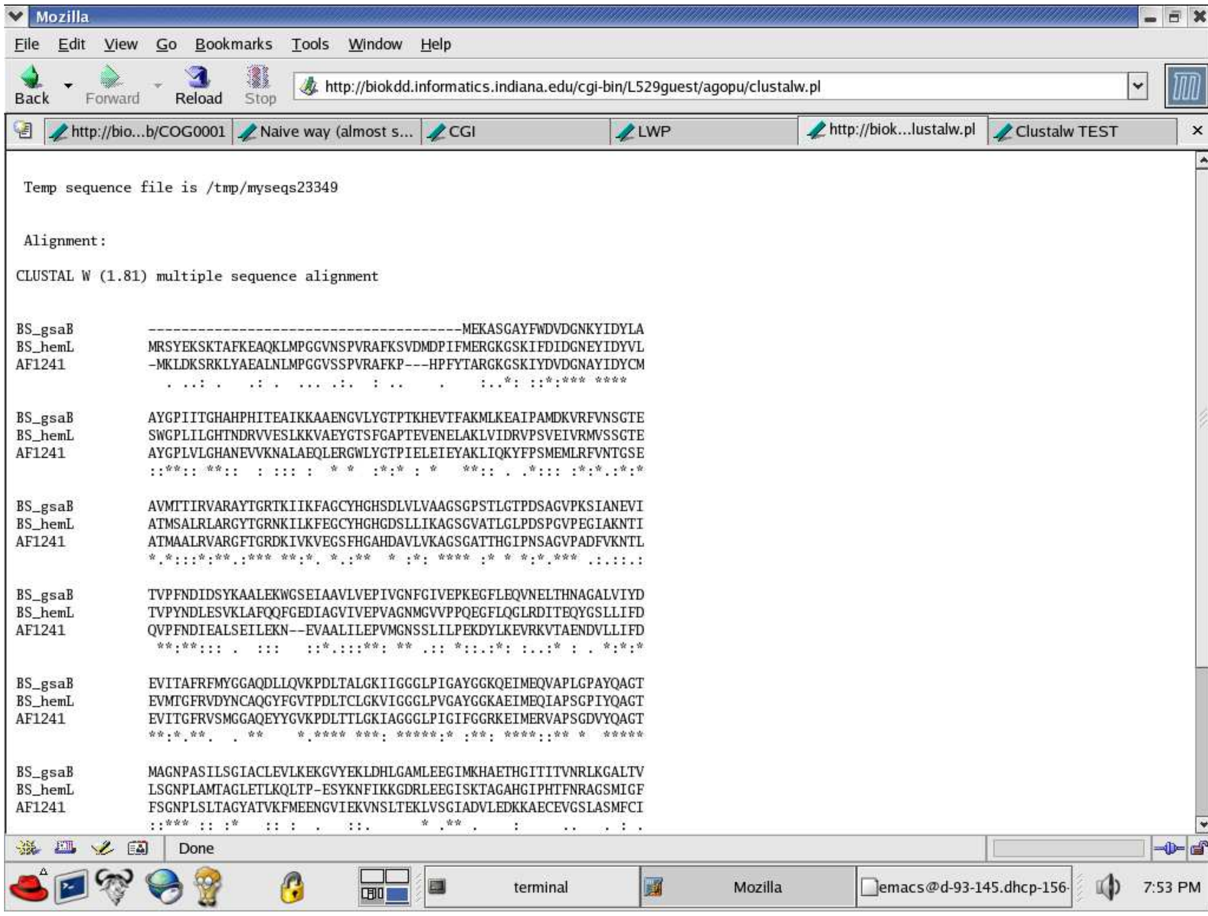
```
>AF1241
MKLDKSRKLYAEALNLMPGGVSSPVRAFKPHPFYTARGKGSKIYDVDGNAY
KNALAEQLERGWLYGTPIELEIEYAKLIQKYFPSMEMLRFVNTGSEATMAA
FHGAHDAVLVKAGSGATTHGIPNSAGVPADFVKNTLQVPFNDIEALSEILI
LPEKDYLKEVRKVTAENDVLLIFDEVITGFRVSMGGAQEYYGVKPDLTTLC
ERVAPSGDVYQAGTFSGNPLSLTAGYATVKFMEENGVIEKVNSLTEKLVSC
MFCIYFGPTPRNYAEALQLNKERFMEFFWRMLENGVFLPPSQYETCFVSFA
>BS_gsaB
MEKASGAYFWDVDGNKYIDYLAAYGPIITGHAHPHITEAIKKAAENGVLYC
MDKVRFVNSGTEAVMTTIRVARAYTGRTKIIKFAGCYHGHSDLVLVAAGSC
```

Figure 1: Screen shot of Clustalw.html

(Replace ‘agopu’ with your user name or whatever you named your directory to be. Remove ‘L529guest’ if you are using your own account.)

Enter some sequence data in the textarea and hit the “Do It” button. This is what you can expect to see:

- Essentially in this example we just passed the sequence through the text field called “atext”.
- We then setup the environment for our process - decide on the TMP dir, where does clustalw exist, so on so forth.
- In the CGI script we collect the data passes along using $q-¿param(-name=¿’atext’). We dump the content of this variable in a file that’s opened in the temporary directory.



```
 Temp sequence file is /tmp/myseqs23349


 Alignment:

CLUSTAL W (1.81) multiple sequence alignment


BS_gsaB         --------------------------------------MEKASGAYFWDVDGNKYIDYLA
BS_hemL         MRSYEKSKTAFKEAQKLMPGGVNSPVRAFKSVDMDPIFMERGKGSKIFDIDGNEYIDYVL
AF1241          -MKLDKSRKLYAEALNLMPGGVSSPVRAFKP---HPFYTARGKGSKIYDVDGNAYIDYCM
                  . ..: .   .: .  ... .:.  : ..     .    :..*: ::*:*** ****

BS_gsaB         AYGPIITGHAHPHITEAIKKAAENGVLYGTPTKHEVTFAKMLKEAIPAMDKVRFVNSGTE
BS_hemL         SWGPLILGHTNDRVVESLKKVAEYGTSFGAPTEVENELAKLVIDRVPSVEIVRMVSSGTE
AF1241          AYGPLVLGHANEVVKNALAEQLERGWLYGTPIELEIEYAKLIQKYFPSMEMLRFVNTGSE
                ::**:: **::  : ::: :  * *  :*:* : *   **:: . .*::: :*:*.:*:*

BS_gsaB         AVMTTIRVARAYTGRTKIIKFAGCYHGHSDLVLVAAGSGPSTLGTPDSAGVPKSIANEVI
BS_hemL         ATMSALRLARGYTGRNKILKFEGCYHGHGDSLLIKAGSGVATLGLPDSPGVPEGIAKNTI
AF1241          ATMAALRVARGFTGRDKIVKVEGSFHGAHDAVLVKAGSGATTHGIPNSAGVPADFVKNTL
                *.*:::*:**.:*** **:*. *.:**  * :*: **** :* * *:*.***  .:.::.:

BS_gsaB         TVPFNDIDSYKAALEKWGSEIAAVLVEPIVGNFGIVEPKEGFLEQVNELTHNAGALVIYD
BS_hemL         TVPYNDLESVKLAFQQFGEDIAGVIVEPVAGNMGVVPPQEGFLQGLRDITEQYGSLLIFD
AF1241          QVPFNDIEALSEILEKN--EVAALILEPVMGNSSLILPEKDYLKEVRKVTAENDVLLIFD
                 **:**::: .  :::   ::*.:::**: ** .:: *::.:*: :..:* : . *:*:*

BS_gsaB         EVITAFRFMYGGAQDLLQVKPDLTALGKIIGGGLPIGAYGGKQEIMEQVAPLGPAYQAGT
BS_hemL         EVMTGFRVDYNCAQGYFGVTPDLTCLGKVIGGGLPVGAYGGKAEIMEQIAPSGPIYQAGT
AF1241          EVITGFRVSMGGAQEYYGVKPDLTTLGKIAGGGLPIGIFGGRKEIMERVAPSGDVYQAGT
                **:*.**.  . **   *.**** ***: *****:* :**: ****::** *  *****

BS_gsaB         MAGNPASILSGIACLEVLKEKGVYEKLDHLGAMLEEGIMKHAETHGITITVNRLKGALTV
BS_hemL         LSGNPLAMTAGLETLKQLTP-ESYKNFIKKGDRLEEGISKTAGAHGIPHTFNRAGSMIGF
AF1241          FSGNPLSLTAGYATVKFMEENGVIEKVNSLTEKLVSGIADVLEDKKAECEVGSLASMFCI
                ::*** :: :*   :: :  .   ::.       * .** .    :      ..   . : .
```

Figure 2: Screen shot of Clustalw.pl script output

- Then we perform clustalw using the system() function and finally read the output file and dump its content into the webpage.

0.3.2 Using Multipart (almost specifically Sequence data)

In this subsection we will check out how we can pass data a bit more elegantly (as data size goes up) from one HTML/CGI script to another. We'll be using a sequence file instead of copying n pasting stuff into a text field. Just create a sequence file with a few sequences and save it as (say) "mysequence". We will try to do clustalw again but this time we will upload the file directly using our browser.

Create a CGI file called "/var/www/cgi-bin/L529guest/agopu/clustalw_file_upload.pl" (as usual replace 'agopu' ....) with the following content. A brief discussion follows the source code.

```
#!/usr/bin/perl

use CGI qw(:standard);

$query = new CGI;
$method="post";
$action="/cgi-bin/L529guest/agopu/clustalw_file_result.pl";

print header, start_html;
print "\n";
print $query->start_multipart_form($method,$action,$encoding);
print "\n";
print filefield(-name=>'upload',-size=>60),br,submit(-label=>'Upload this file');
print "\n";
print $query->endform;
print "\n";
print end_html;
```

Create another CGI script called “/var/www/cgi-bin/L529guest/agopu/clustalw_file_result.pl” (as usual replace ‘agopu’ ....) with the following content.

```
#!/usr/bin/perl

use CGI qw(:standard);

$file = param('upload');

print header, start_html;
print "<pre>";

my      $CLUSTALW="/usr/local/bio/bin/clustalw";
my      $DIR="/tmp/";
my      $TMPSEQS=$DIR."myseqs".$$;

$|=1;
print " Temp FILENAME = $TMPSEQS<br>";
print " FILENAME Received = $file<br> Contents: <br>";
open(THISF, ">$TMPSEQS") or die "open $TMPSEQS failed";
while(<$file>) {
        print THISF  $_;
        print $_;
}
close THISF;

system ("$CLUSTALW $TMPSEQS > $TMPSEQS.out 2>&1");
print " <p> Alignment: <br>";
open(THISF, "$TMPSEQS.aln") or die "open $TMPSEQS.aln failed";
```

```
while(<THISF>) {
        print $_;
}
close THISF;
print end_html;
```

Do not forget to change permissions for these files as shown in the previous section:

```
[L529guest@biokdd agopu]\$ chmod 755 /var/www/cgi-bin/L529guest/agopu/*.pl
```

This is the stuff you can expect to see on your browser if you visit:

http://biokdd.informatics.indiana.edu/cgi-bin/L529guest/agopu/clustalw_file_upload.pl

(Replace 'agopu' with your user name or whatever you named your directory to be. Remove 'L529guest' if you are using your own account.)

Upload a sequence file using the Browse button and the Upload button. This is what you can expect to see as results of your script.

- In this example we just pass the sequence file using the upload multipart thingy - it's named "upload".
- We then setup the environment for our process - decide on the TMP dir, where does clustalw exist, so on so forth.
- In the CGI script we collect the data passes along using $file = param('upload'). We dump the content of this variable in a file that's opened in the temporary directory using a while file reader loop.
- Then we perform clustalw using the system() function and finally read the output file and dump its content into the webpage as before.

0.4 GNUPlot Basics

In this section we will check out how GNUPlot can be used to make simple plots of datasets. Of course there is a lot more it's capable of, but we won't be going into much detail. Refer one of the ubiquitous tutorials/manuals available on the web. There are a couple of links Sun has on the class homepage [1].

The stuff below is modified from Sun Kim's example page showing the use of GNUPlot.

- Prepare data file

```
[L529guest@biokdd agopu]\$cat > data1
1  10
```

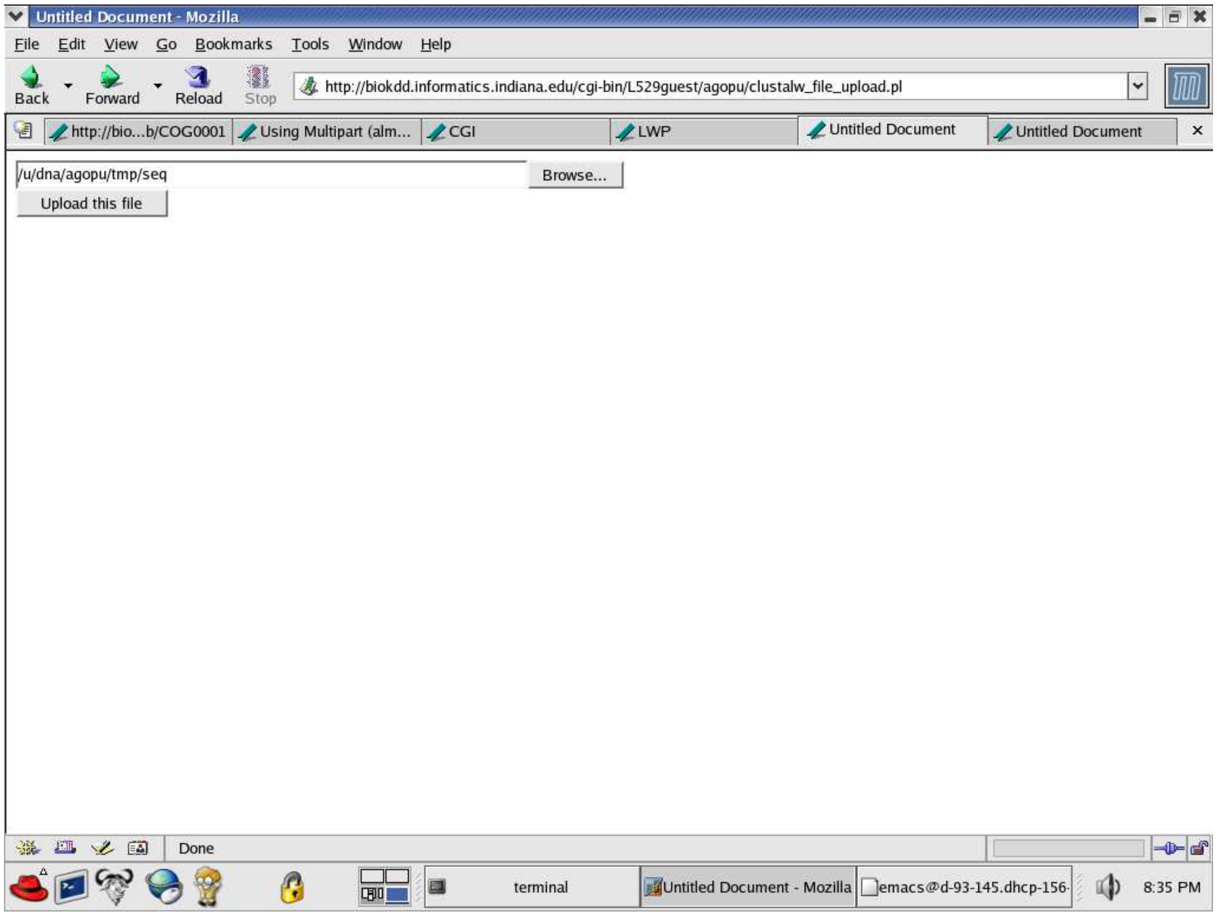


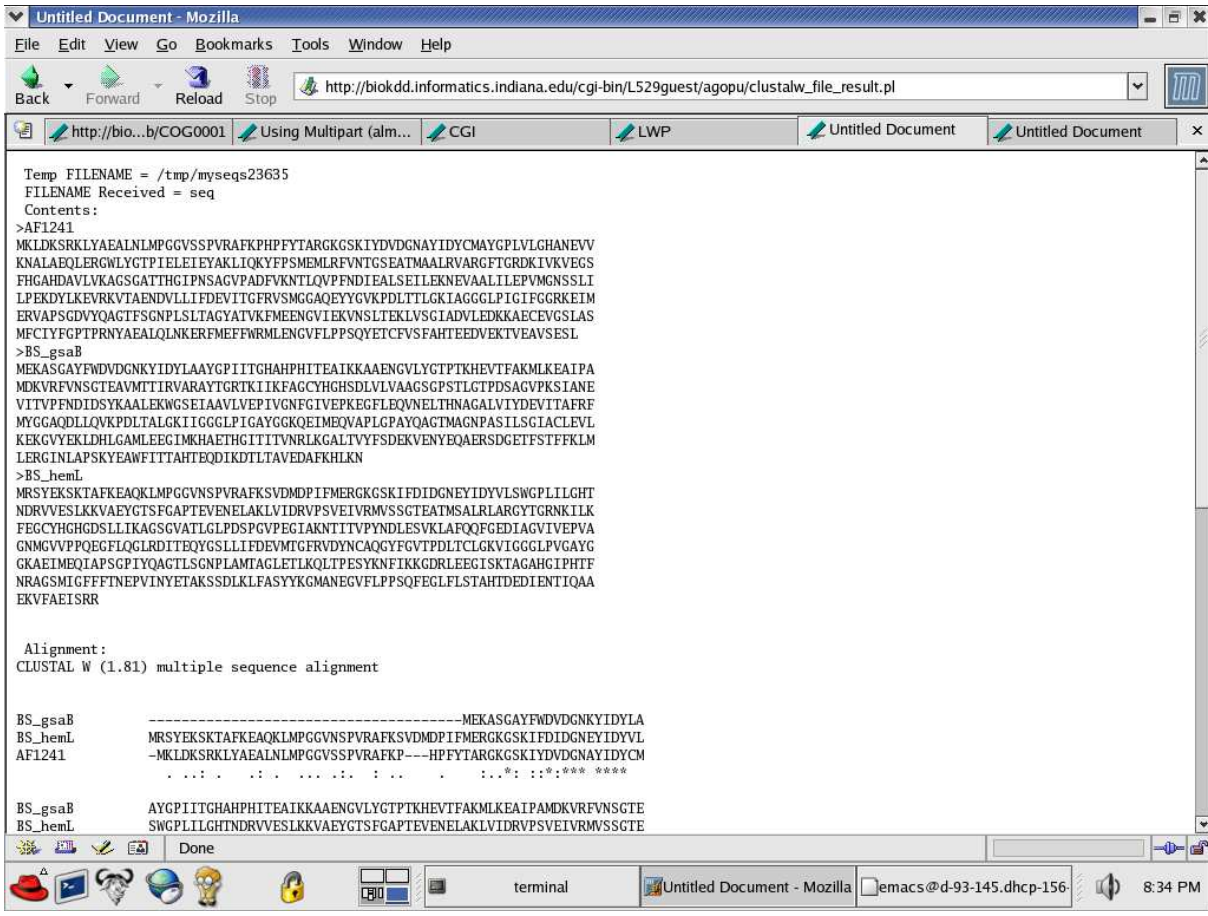
Figure 3: Screen shot of Clustalw.html using Multipart

```
5  15
10 20
```

Do Ctrl + D to quit.

- GNUPlot on Unix shell prompt:

```
[L529guest@biokdd agopu]\$ gnuplot
        G N U P L O T
        Version 3.7 patchlevel 2
        last modified Sat Jan 19 15:23:37 GMT 2002
        System: Linux 2.4.18-14smp
```



Untitled Document - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://biokdd.informatics.indiana.edu/cgi-bin/L529guest/agopu/clustalw_file_result.pl
http://bio...b/COG0001
Using Multipart (alm...
CGI
LWP
Untitled Document
Untitled Document
Done
terminal
Untitled Document - Mozilla
emacs@d-93-145.dhcp-156
8:34 PM

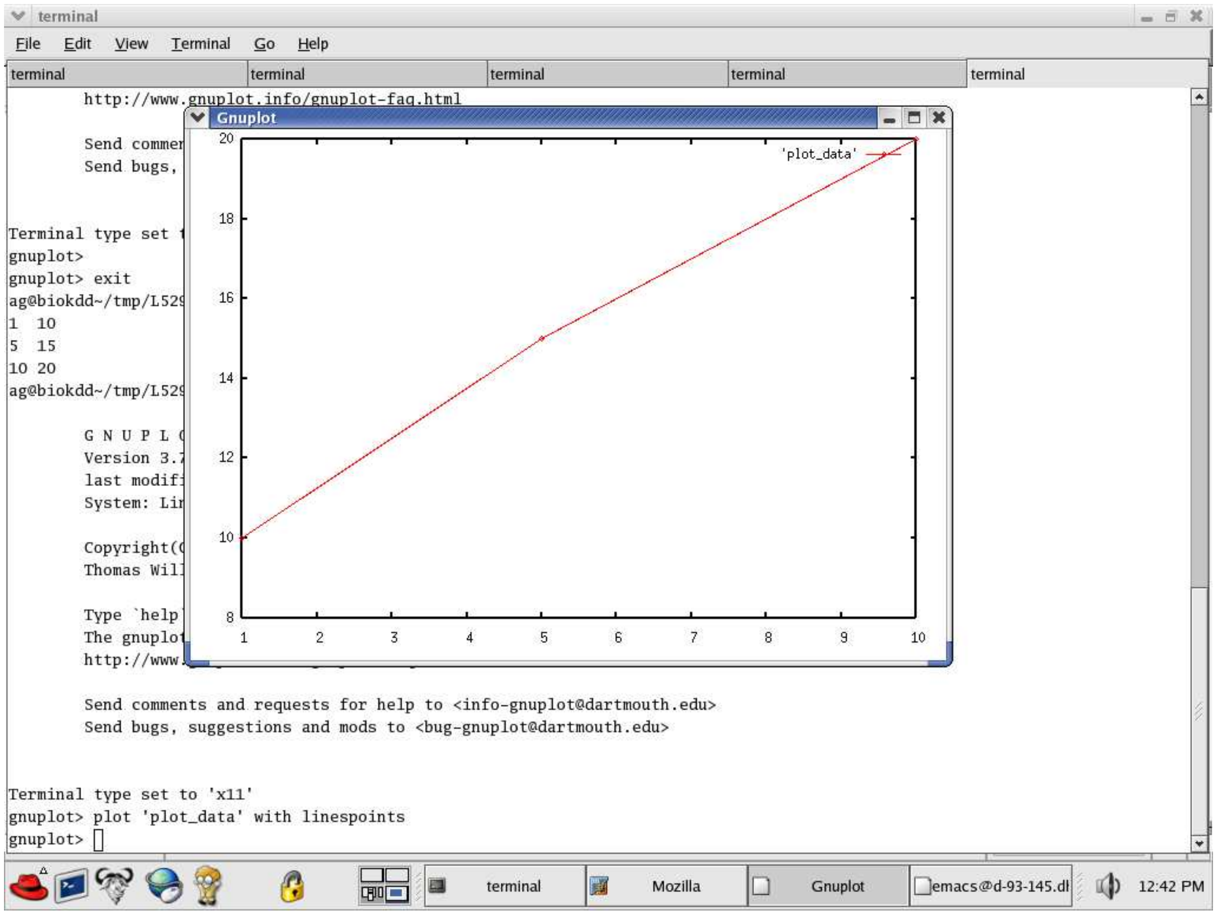
```
 Temp FILENAME = /tmp/myseqs23635
 FILENAME Received = seq
 Contents:
>AF1241
MKLDKSRKLYAEALNLMPGGVSSPVRAFKPHPFYTARGKGSKIYDVDGNAYIDYCMAYGPLVLGHANEVV
KNALAEQLERGWLYGTPIELEIEYAKLIQKYFPSMEMLRFVNTGSEATMAALRVARGFTGRDKIVKVEGS
FHGAHDAVLVKAGSGATTHGIPNSAGVPADFVKNTLQVPFNDIEALSEILEKNEVAALILEPVMGNSSLI
LPEKDYLKEVRKVTAENDVLLIFDEVITGFRVSMGGAQEYYGVKPDLTTLGKIAGGGLPIGIFGGRKEIM
ERVAPSGDVYQAGTFSGNPLSLTAGYATVKFMEENGVIEKVNSLTEKLVSGIADVLEDKKAECEVGSLAS
MFCIYFGPTPRNYAEALQLNKERFMEFFWRMLENGVFLPPSQYETCFVSFAHTEEDVEKTVEAVSESL
>BS_gsaB
MEKASGAYFWDVDGNKYIDYLAAYGPIITGHAHPHITEAIKKAAENGVLYGTPTKHEVTFAKMLKEAIPA
MDKVRFVNSGTEAVMTTIRVARAYTGRTKIIKFAGCYHGHSDLVLVAAGSGPSTLGTPDSAGVPKSIANE
VITVPFNDIDSYKAALEKWGSEIAAVLVEPIVGNFGIVEPKEGFLEQVNELTHNAGALVIYDEVITAFRF
MYGGAQDLLQVKPDLTALGKIIGGGLPIGAYGGKQEIMEQVAPLGPAYQAGTMAGNPASILSGIACLEVL
KEKGVYEKLDHLGAMLEEGIMKHAETHGITITVNRLKGALTVYFSDEKVENYEQAERSDGETFSTFFKLM
LERGINLAPSKYEAWFITTAHTEQDIKDTLTAVEDAFKHLKN
>BS_hemL
MRSYEKSKTAFKEAQKLMPGGVNSPVRAFKSVDMDPIFMERGKGSKIFDIDGNEYIDYVLSWGPLILGHT
NDRVVESLKKVAEYGTSFGAPTEVENELAKLVIDRVPSVEIVRMVSSGTEATMSALRLARGYTGRNKILK
FEGCYHGHGDSLLIKAGSGVATLGLPDSPGVPEGIAKNTITVPYNDLESVKLAFQQFGEDIAGVIVEPVA
GNMGVVPPQEGFLQGLRDITEQYGSLLIFDEVMTGFRVDYNCAQGYFGVTPDLTCLGKVIGGGLPVGAYG
GKAEIMEQIAPSGPIYQAGTLSGNPLAMTAGLETLKQLTPESYKNFIKKGDRLEEGISKTAGAHGIPHTF
NRAGSMIGFFFTNEPVINYETAKSSDLKLFASYYKGMANEGVFLPPSQFEGLFLSTAHTDEDIENTIQAA
EKVFAEISRR


 Alignment:
CLUSTAL W (1.81) multiple sequence alignment


BS_gsaB         --------------------------------------MEKASGAYFWDVDGNKYIDYLA
BS_hemL         MRSYEKSKTAFKEAQKLMPGGVNSPVRAFKSVDMDPIFMERGKGSKIFDIDGNEYIDYVL
AF1241          -MKLDKSRKLYAEALNLMPGGVSSPVRAFKP---HPFYTARGKGSKIYDVDGNAYIDYCM
                  . ..: .   .: .  ... .:.  : ..    .    :..*: ::*:*** ****

BS_gsaB         AYGPIITGHAHPHITEAIKKAAENGVLYGTPTKHEVTFAKMLKEAIPAMDKVRFVNSGTE
BS_hemL         SWGPLILGHTNDRVVESLKKVAEYGTSFGAPTEVENELAKLVIDRVPSVEIVRMVSSGTE
```

Figure 4: Screen shot of Clustalw.pl script output using Multipart

```
          Copyright(C) 1986 - 1993, 1998 - 2002
          Thomas Williams, Colin Kelley and many others

          Type `help` to access the on-line reference manual
          The gnuplot FAQ is available from
          http://www.gnuplot.info/gnuplot-faq.html

          Send comments and requests for help to <info-gnuplot@dartmouth.edu>
          Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>


Terminal type set to 'x11'
```

```
gnuplot> plot 'data1' with linespoints
gnuplot> quit
[L529guest@biokdd agopu]\$
```

NOTE: When I tried this on the 'L529guest' account, I ran into problems with X window emulation for some reason. If it does not work for you, just see below for a sample screenshot of what you would have got if it did work! The rest of the stuff – creating image/EPS files should work for everyone, I am sure!



```
        http://www.gnuplot.info/gnuplot-faq.html

        Send commen
        Send bugs,


Terminal type set t
gnuplot>
gnuplot> exit
ag@biokdd~/tmp/L529
1  10
5  15
10 20
ag@biokdd~/tmp/L529

        G N U P L (
        Version 3.7
        last modifi
        System: Lin

        Copyright((
        Thomas Will

        Type `help`
        The gnuplot
        http://www.

        Send comments and requests for help to <info-gnuplot@dartmouth.edu>
        Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>


Terminal type set to 'x11'
gnuplot> plot 'plot_data' with linespoints
gnuplot> 
```

Figure 5: Screen shot - Unix usage of GNUPlot

- Using command file for gnuplot

```
[L529guest@biokdd agopu]\$ cat cmd-file
set term postscript mono "Times-Roman" 24
set xrange [0:20]
set xlabel "residue position"
set ylabel "relative entropy"
set output "plot_out.ps"
plot 'data2' title "Relative Entropy Plot"
[L529guest@biokdd agopu]\$
```

Then:

```
[L529guest@biokdd agopu]\$ gnuplot < cmd-file
[L529guest@biokdd agopu]\$ ghostview plot_out.ps
```

- Dynamically creating graph images

```
[L529guest@biokdd agopu]\$ cat cmd-file2
set term png mono medium
set xrange [0:20]
set xlabel "Residue Position"
set ylabel "Relative Entropy"
set output "plot_out.png"
plot 'data2' title "Relative Entropy Plot" with linespoints
[L529guest@biokdd agopu]\$
```

Then:

```
[L529guest@biokdd agopu]\$ gnuplot < cmd-file2
[L529guest@biokdd agopu]\$ gimp plot_out.png
```

Just a couple of notes about the commands used:

- 'plot' is the primary command to collect the data points and plot them
- The 'with' keyword is used to specify how you want the plot to look like – you want just the points plotted? Or do you want lines to connect the points?

- ‘set’ is used to specify a whole bunch of parameters, a few of which are shown in the above code.
  - ‘term’ is used to specify the output medium – the screen? (default), a PostScript file? ....
  - ‘output’ is used to specify an output file name (used in conjunction with ‘set term ....’
  - ‘xrange’, ‘xlabel’ and ‘ylabel’ are kinda self explanatory.
- Use ghostview or ggv - whichever is installed to open PS files!
- Using GIMP might require you to go through a couple of annoying installation steps!

0.4.1 Using GNUPlot inside CGI scripts

In this section we will see how we can create a plot inside a CGI script and output the resulting image

```
[L529guest@biokdd agopu]\$ cat > data_show1
1 0.1
2 0.2
3 0.3
4 0.4
5 0.5
[L529guest@biokdd agopu]\$ cat > data_show2
1 1.1
2 1.2
3 1.3
4 1.4
5 1.5
[L529guest@biokdd agopu]\$ chmod 644 data*

[L529guest@biokdd agopu]\$ cat > plot_show.pl
#!/usr/bin/perl

$|=1;
print "Content-type: text/html\n\n";

my $filename="/tmp/anyname". $$. ".png";  # use $$ to make the file name is unique.
umask(022);  # to allow read and execute
print "<img src=\"$filename\">";

# remember multiple lines should end with two backslashes, \\
open(G, "|gnuplot") or die "gnuplot failed";
print G <<ENDOFHERE
set term png color medium
set xrange [0:20]
set xlabel "Residue Position"
set ylabel "Relative Entropy"
set output '$filename'
plot 'data_show1' title "Group1" with linespoints, \\
'data2_show' title "Group2" with linespoints
ENDOFHERE

[L529guest@biokdd agopu]\$ chmod 755 *.pl
```

This is the stuff you can expect to see on your browser if you visit:
http://biokdd.informatics.indiana.edu/cgi-bin/L529guest/agopu/plot_show.pl
(Replace 'agopu' with your user name or whatever you named your directory to be. Remove 'L529guest' if you are using your own account.)
Just some notes about the above shown code:

- Most of the GNUPlot code is essentially the same as the ones used in the previous subsection
- Note that we redirect the plotting code directly to gnuplot – that's the reason we've used *open(G, "—gnuplot")*
- Also note use of 'umask' to set the default file permissions to 755 (the compliment of 022!)
- If you have not noticed this thus far, we use $$ in filenames to keep them unique. $$ gives the process id for the process spawned by the script execution

0.5 LWP

```
[L529guest@biokdd agopu]\$ cat > lwp.pl
 #!/usr/bin/perl

 # Create a user agent object
 use LWP::UserAgent;

 use CGI;
 $query = new CGI;

 print $query->header;

 $ua = LWP::UserAgent->new;
 $ua->agent("MyApp/0.1 ");
```

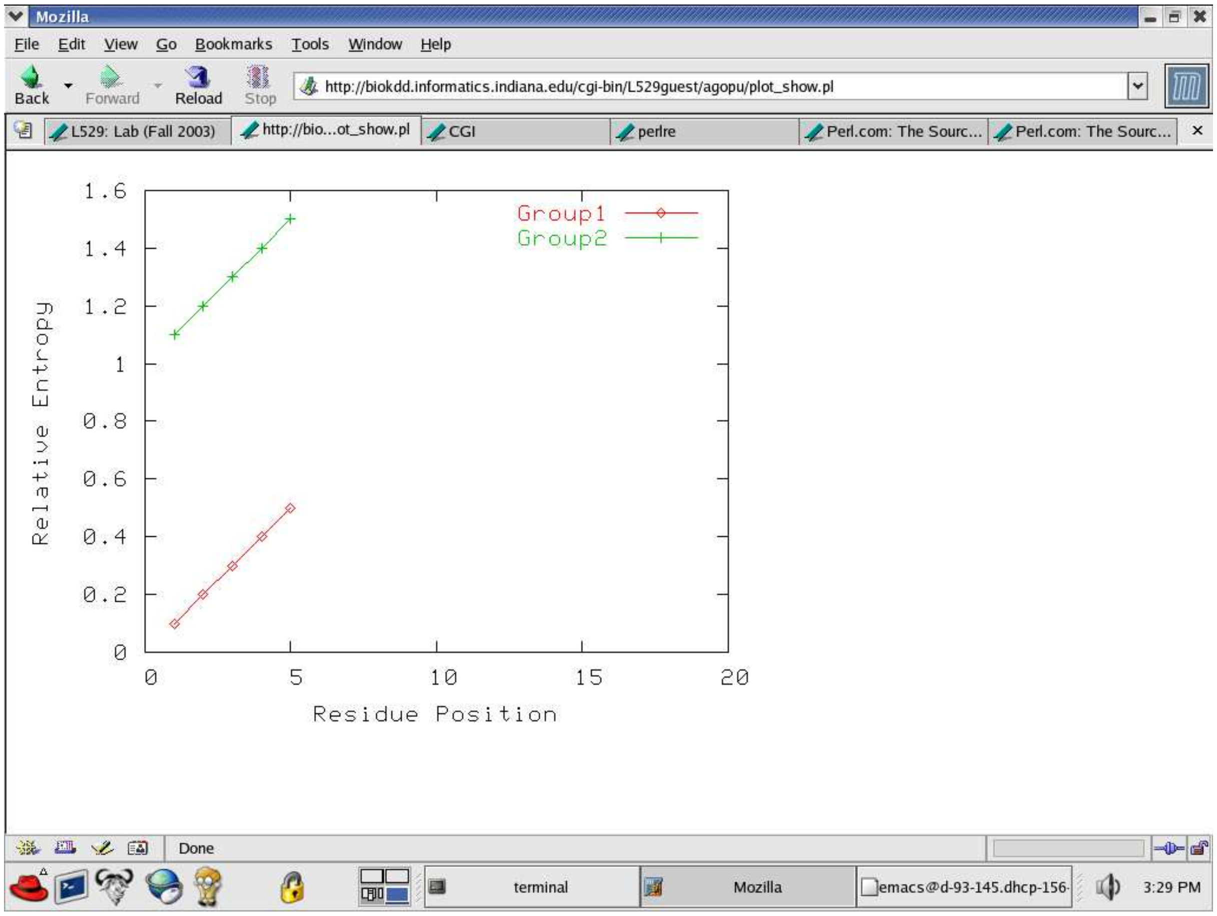


Figure 6: Screen shot of GNUPlot output when called within a CGI script

```
 # Create a request
 my $req = HTTP::Request->new(POST => 'http://www.perl.com/');
 $req->content_type('application/x-www-form-urlencoded');
# $req->content('match=www&errors=0');

 # Pass request to the user agent and get a response back
 my $res = $ua->request($req);

 # Check the outcome of the response
 if ($res->is_success) {
     print $res->content;
 } else {
     print "Bad luck this time\n";
 }

 print $query->end_html;

[L529guest@biokdd agopu]$ chmod 755 *.pl
```

If things go according to plan, you should be redirected to www.perl.com on your browser if you visit:

http://biokdd.informatics.indiana.edu/cgi-bin/L529guest/agopu/lwp.pl

(Replace 'agopu' with your user name or whatever you named your directory to be. Remove 'L529guest' if you are using your own account.)

Ofcourse what has been abstracted out of this dicussion is that if the server (www.per.com in this case) refused a connection (say, because we had connected less than 'n' secs back) then the script would have printed "Bad luck this time" instead of taking you to per.com. That is the whole objective (in this example, that is; LWP has a lot more capabilities) of using LWP – to prevent hoggin a server with a deluge of requests. We can setup the script to send requests, for instance, once every 'n' seconds. That is left as an exercise!

Hint: Try using "sleep n-seconds' in your CGI script

OR

See here: http://bio.informatics.indiana.edu/L529/lab/lwp.pl.sample

0.6 Graph Packages

In this section we'll look at a graph package that can be used to construct simple graphs.

Before going into graphs, there's a quick overview of the 'Set' package.

```
[L529guest@biokdd agopu]\$ cat > set.pl
#!/usr/bin/perl

use Set::Scalar;

my $A = Set::Scalar->new('a', 'b', 'c');
my $B = Set::Scalar->new('b', 'c');
my $C = Set::Scalar->new('c', 'd');

print "INTERSECT of A, B, C = ", $A->intersection($B, $C), "\n";
print "UNION of A, B, C = ", $A->union($B, $C), "\n";
```

In the above code, we've just defined three sets A, B and C with the various elements as shown. Also use of simple operators like intersection() and union()

is shown. The output of this script follows the source code of two other scripts (showing graph stuff).

The following two source code snippets show how graphs can be constructed.

```
[L529guest@biokdd agopu]\$ cat > graph1.pl
#!/usr/bin/perl

use Graph::Undirected;
use Graph::DFS;

my $g = Graph::Undirected->new;

$g->add_edge('a', 'b');
$g->add_edge('a', 'c');
$g->add_edge('a', 'd');
$g->add_edge('d', 'e');
$g->add_edge('f', 'g');


$dfs = Graph::DFS->new($g);

@RA  = $dfs->roots;
print $dfs->roots, "\n";


%R = $dfs->vertex_roots;

pr_root('a');
pr_root('b');
pr_root('f');

sub
pr_root
{
        print "root of $_[0] is ",  $R{$_[0]}, "\n";
        print "root of $_[0] is ",  $RA[$R{$_[0]}], "\n";
}
```

```
[L529guest@biokdd agopu]\$ cat > graph2.pl
#!/usr/bin/perl

use Set::Scalar;
use Graph::Undirected;
use Graph::DFS;

my $g = Graph::Undirected->new;

$g->add_weighted_edge('a', 1.0, 'b');
$g->add_weighted_edge('a', 1e-1, 'c');
$g->add_weighted_edge('a', 1e-4, 'd');
$g->add_weighted_edge('d', 1e-2, 'e');
$g->add_weighted_edge('f', 1e-3, 'g');

print "The density of the graph is ", $g->density, "\n";

my $allvset = Set::Scalar->new($g->vertices);

foreach $v ($g->vertices) {
        my @ne = $g->neighbors($v);
        my $vset = Set::Scalar->new(@ne);

        $g->set_attribute($v, $vset);
        print "The neighbors of $v are @ne \n";
}

#see if how many vertices cover the shole graph
$vcover = Set::Scalar->new;
foreach $v ($g->vertices) {
        my $vset = Set::Scalar->new(qw(@ne));

        $g->set_attribute($v, $vset);
        print "The neighbors of $v are @ne \n";
}

#see if how many vertices cover the shole graph
$vcover = Set::Scalar->new;
foreach $v ($g->vertices) {
        my $vset = Set::Scalar->new(qw(@ne));

        my $thisset = $g->get_attribute($v);

        my $setdiff = $allvset->difference($thisset);
        $setdiff->delete($v);
        my @notcovered = $setdiff->members;
        if (defined(@notcovered)) {
                print "Vertices [ @notcovered ] are not covered by $v \n";
        }

        $vcover->union($thisset);
        if ($vcover->is_equal($allvset)) {
                print "All vertices are covered by nodes ";
        }
}

$SP = $g->APSP_Floyd_Warshall($v);
#print "Floyd-Warshall graph ", $SP, "\n";
$path = $SP->get_attribute("path", 'a', 'e');
print " The shortest path from a to e is @$path \n";
```

This is what you can expect to see as output. Do note that these scripts don't spit HTML stuff as part of the output - no CGI object exists either! Yeah! these are essentially perl scripts. It's left as an exercize for you to modify the scripts to make them web-accessible. By now you should be able to do that, otherwise this whole exercize would be one in futility!

Anyway here we go ...

```
[L529guest@biokdd agopu]\$
[L529guest@biokdd agopu]\$ chmod 755 *.pl
[L529guest@biokdd agopu]\$
[L529guest@biokdd agopu]\$ perl set.pl
INTERSECT of A, B, C = (c)
UNION of A, B, C = (a b c d)

[L529guest@biokdd agopu]\$ perl graph1.pl
af
root of a is 0
root of a is a
root of b is 0
root of b is a
root of f is 1
root of f is f

[L529guest@biokdd agopu]\$ perl graph2.pl
The density of the graph is 0.238095238095238
The neighbors of a are b c d
The neighbors of b are a
The neighbors of c are a
The neighbors of d are e a
The neighbors of e are d
The neighbors of f are g
The neighbors of g are f
The neighbors of a are
The neighbors of b are
The neighbors of c are
The neighbors of d are
The neighbors of e are
The neighbors of f are
The neighbors of g are
Vertices [ e c b g d f ] are not covered by a
Vertices [ e c g d f ] are not covered by b
Vertices [ e g d f ] are not covered by c
Vertices [ e g f ] are not covered by d
Vertices [ g f ] are not covered by e
Vertices [ g ] are not covered by f
Vertices [  ] are not covered by g
All vertices are covered by nodes  The shortest path from a to e is a d e
```

Bibliography

[1] L529 *http://bio.informatics.indiana.edu/L529*.

[2] Perldocs: CGI Section *http://www.perldoc.com/perl5.8.0/lib/CGI.html*.

[3] Perldocs: LWP Section *http://www.perldoc.com/perl5.8.0/lib/LWP.html*.

[4] L529 Gnuplot example *http://bio.informatics.indiana.edu/L529/lab/gnuplot.example.html* .

[5] Tutorial on CGI, etc *http://www.extropia.com/tutorials/perl_cgi/pre_requisite_intro.html*.