Gnuplot as a CARMEN Service

Wrapping Gnuplot as a service raises several problems, the biggest of which is passing the input parameters into the Gnuplot code (i.e. input and output filenames). However, several solutions can be realised by embedding the Gnuplot commands into Perl or bash scripts.

Note that these solutions create specific Gnuplot services based upon a specific Gnuplot script, rather than a generic Gnuplot service.

Perl Gnuplot Solution

We can realise a Gnuplot service by embedding Gnuplot code into a Perl script and execute the Gnuplot command through a pipe. It is also possible to use Perl Gnuplot libraries, but these do not always keep pace with the main Gnuplot application. Once we have gnuplot embedded in a perl script it can be easily wrapped up into CARMEN service and deployed on the CARMEN Portal.

So, how do we call Gnuplot from Perl? The following example Perl script template is all we need. In this example, the script takes three input arguments and has one output. The inputs are; two input filenames containing the data for Gnuplot to use, and the name of the output file you wish to generate. The single output is the name of the generated file (if successful).

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

# check that the number of arguments is correct
my $numArgs = $#ARGV + 1;
if ($numArgs != 3 ) {
        print "\tincorrect number of arguments", "\n";
        print "\tusage:\t inputFilename inputFilename outputFilename\n";
        exit(-1);
}

# process imput arguments, where ARGV[i] is an array containing the command
line parameters
my $inputFile1 = $ARGV[0];
my $inputFile2 = $ARGV[1];
my $outputFile = $ARGV[2];

# Embed gnuplot and code
open (GNUPLOT, "|gnuplot");
print GNUPLOT <<GnuplotCommands;
   ... add gnuplot code here ...
GnuplotCommands
close(GNUPLOT);

# check for gnuplot success (i.e. 0)
if ($? != 0){exit(1);}
# set service output (a comma seperated string if more than one output)
print "<output>", $outputFilename, "</output>", "\n";
exit(0);
```

Your Gnuplot code can be pasted into the script where marked "... add Gnuplot code here . . .". Any explicitly named filenames in the Gnuplot code must be replaced with $inputFile, $outputFile, etc.

Note: the number of inputs and outputs is completely up to the application developer. Just modify the example to suit your own needs.

The following is a worked example, that takes a Gnuplot script that plots a graph to a png file, and embeds it into a perl script.

```
set ylabel "time (in seconds)"
set xlabel "k"
set term png
set output "output.png"
plot [:][0:] "times.txt" using 1:2 with linespoints, "times.txt" using 1:3
with linespoints
```

If we save the above Gnuplot script to a file called ***demo.gp***, and have a data file called ***times.tx*t** (shown below) in the same directory.

```
5       26.53   9.01
10      27.12   9.74
15      28.31   10.71
20      29.30   11.78
25      30.13   12.85
30      31.28   13.80
35      32.27   14.90
40      33.12   15.73
45      33.99   16.63
50      35.48   17.43
```

We can generate the graph using

```
gnuplot demo.gp
```

To embed the Gnuplot script into a Perl script, we need to replace "times.txt" with the placeholder "$inputFile" and "output.png" with the placeholder "$outputFile". This can then be pasted directly into the Perl script template shown above. i.e.

```
my $numArgs = $#ARGV + 1;
if ($numArgs != 2 ) {
        print "\tincorrect number of arguments", "\n";
        print "\tusage:\t inputFilename outputFilename\n";
        exit(-1);
}

# process imput arguments
my $inputFile = $ARGV[0];
my $outputFile = $ARGV[1];

# Embed gnuplot and code
open (GNUPLOT, "|gnuplot");
print GNUPLOT <<GnuplotCommands;
set ylabel "time (in seconds)"
set xlabel "k"
set term png
```

```
set output "$outputFile"
plot [:][0:] "$inputFile" using 1:2 with linespoints, "$inputFile" using
1:3 with linespoints
GnuplotCommands
close(GNUPLOT);
if ($? != 0){exit(1);}
print "<output>", $outputFile, "</output>", "\n";
exit(0);
```

Once this script is saved to a file (i.e. demo.perl), it can be executed using the following command line

```
perl demo.perl demo.txt output.png
```

The resultant plot should exist in the graphics file ***output.png*** in the current working directory.

This Perl script can now be wrapped into a CARMEN service using the CARMEN Portal.

BASH Gnuplot Solution

A similar approach is achieved using bash scripts under Unix-type systems. A script is constructed using a variable, and then this is piped to the gnuplot command.

For the same example Gnuplot script shown above, here is the required bash script

```
#!/bin/bash
# set up gnuplot
source /wrg/profile.d/gnuplot

no_of_expected_args=2
if [ $# -ne $no_of_expected_args ]; then
        echo "usage: $0 <input-file> <output-PNG-file>"
        exit
fi
inputFile="$1"
outputFile=$2

# put the gnulpot commands in a variable
plot_cmd="set ylabel 'time (in seconds)';"
plot_cmd="$plot_cmd set xlabel 'k';"
plot_cmd="$plot_cmd set term png;"
plot_cmd="$plot_cmd set output '$outputFile';"
plot_cmd="$plot_cmd plot [:][0:] '$inputFile' using 1:2 with linespoints, '$inputFile' using 1:3 with
linespoints;"
# pipe the command to gnuplot
echo $plot_cmd | gnuplot
# error handling
if [ $? -ne 0 ]; then
   exit -1
fi
# inform the carmen system of the output file
echo "<output>"$outputFile"</output>"
```

If the above bash script is saved as “demo.bash”, it can be executed on the command line using the following;

```
demo.bash  times.txt  output.png
```