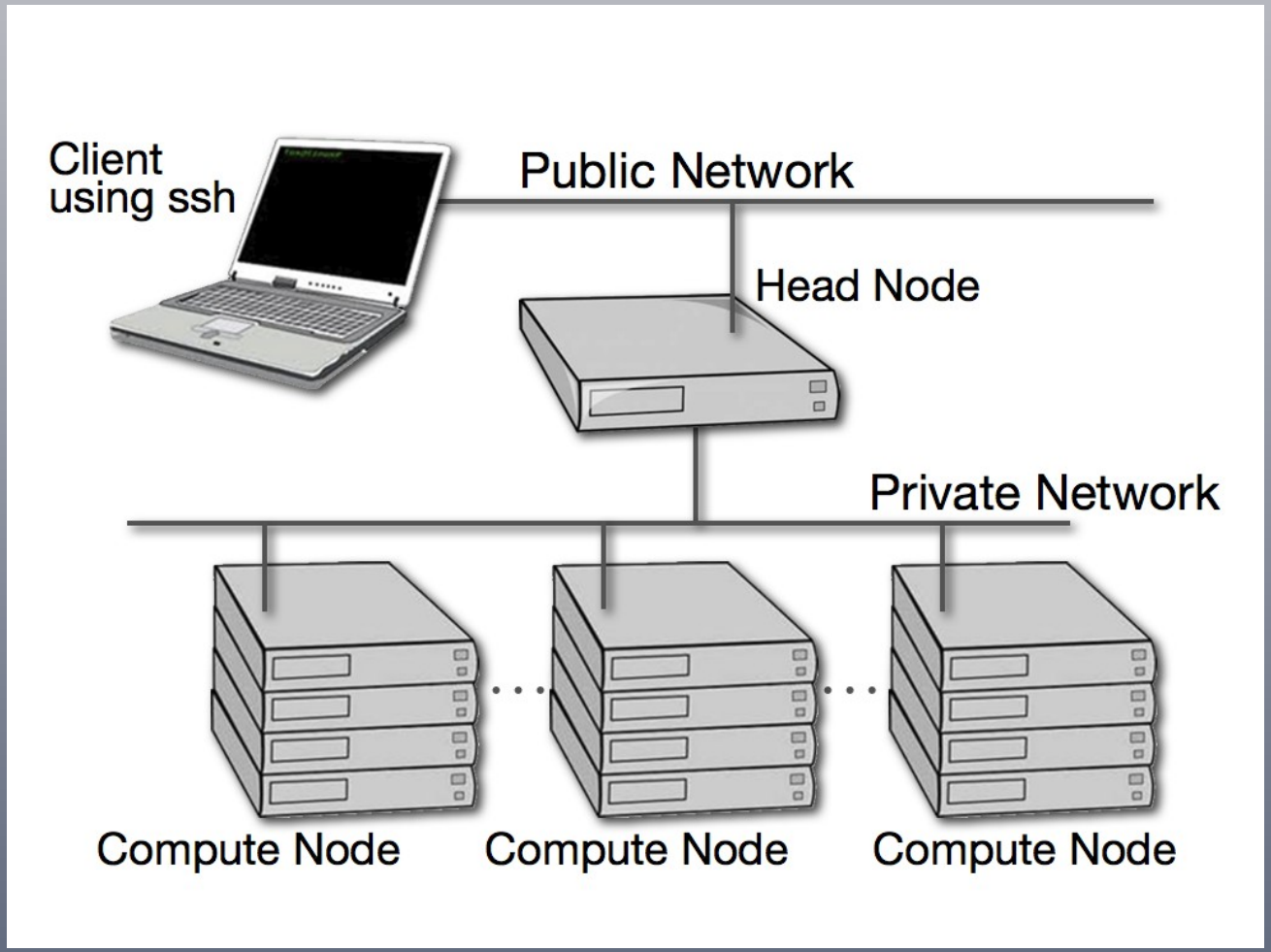
Part II

UNIX Workshop Series: Shell Programming

Objectives

- Overview – Connecting with ssh
- Bash Shell
- Script Basics
- Script Project
  - This project is based on using the gnuplot program which reads a command file, a data file and writes an image file as an x-y plot.

Overview

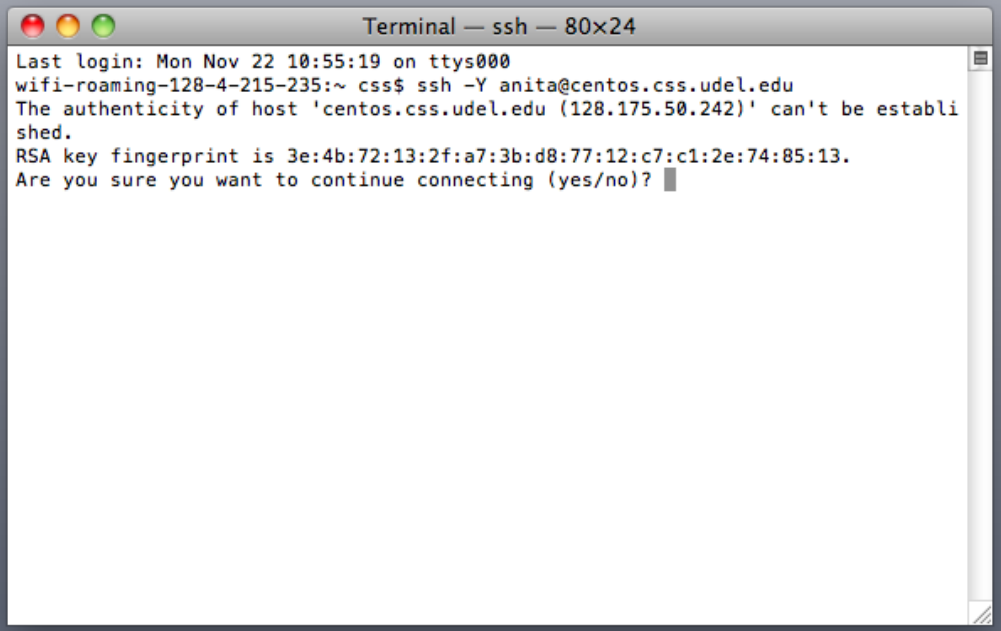

Client
using ssh
Public Network
Head Node
Private Network
Compute Node
Compute Node
Compute Node

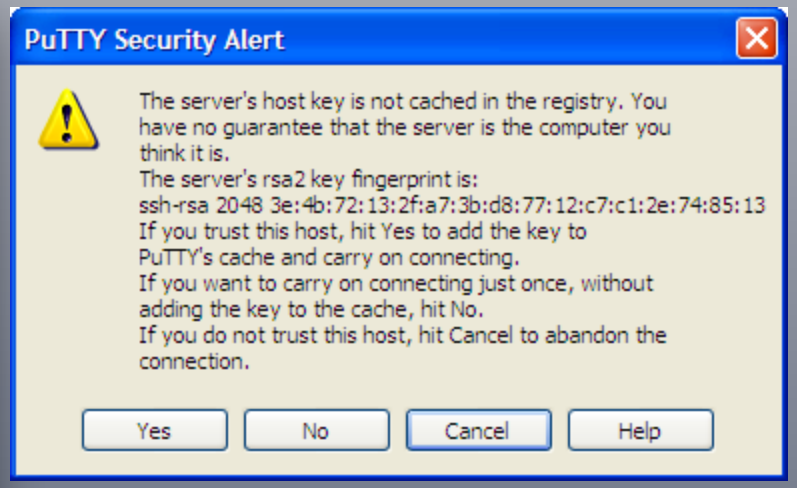
Connecting with ssh

- Open a Terminal program
  - Mac: ***Applications*** **>** ***Utilities*** **>** ***Terminal***
    `ssh –Y` **`username`**`@centos.css.udel.edu`
  - Linux: In local shell
    `ssh –Y` **`username`**`@centos.css.udel.edu`
  - Windows: Start Xming and PuTTY
    Create a saved session for the remote host name `centos.css.udel.edu` using **`username`**

Connecting with ssh

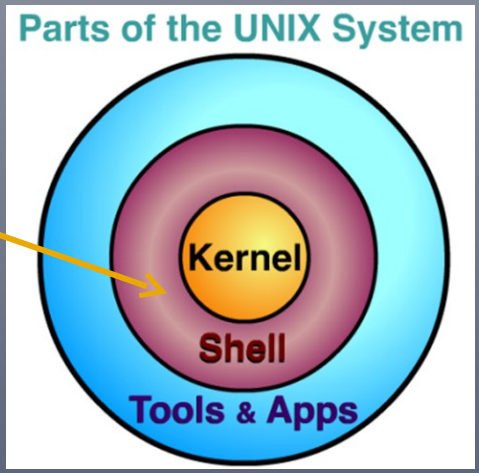
- First time you connect





Shell Basics

- The shell is a command interpreter. We are using the bash shell (/bin/bash).
- It is the insulating layer between the operating system kernel and the user.
- It is also a powerful programming language.
- A shell program is called a scri

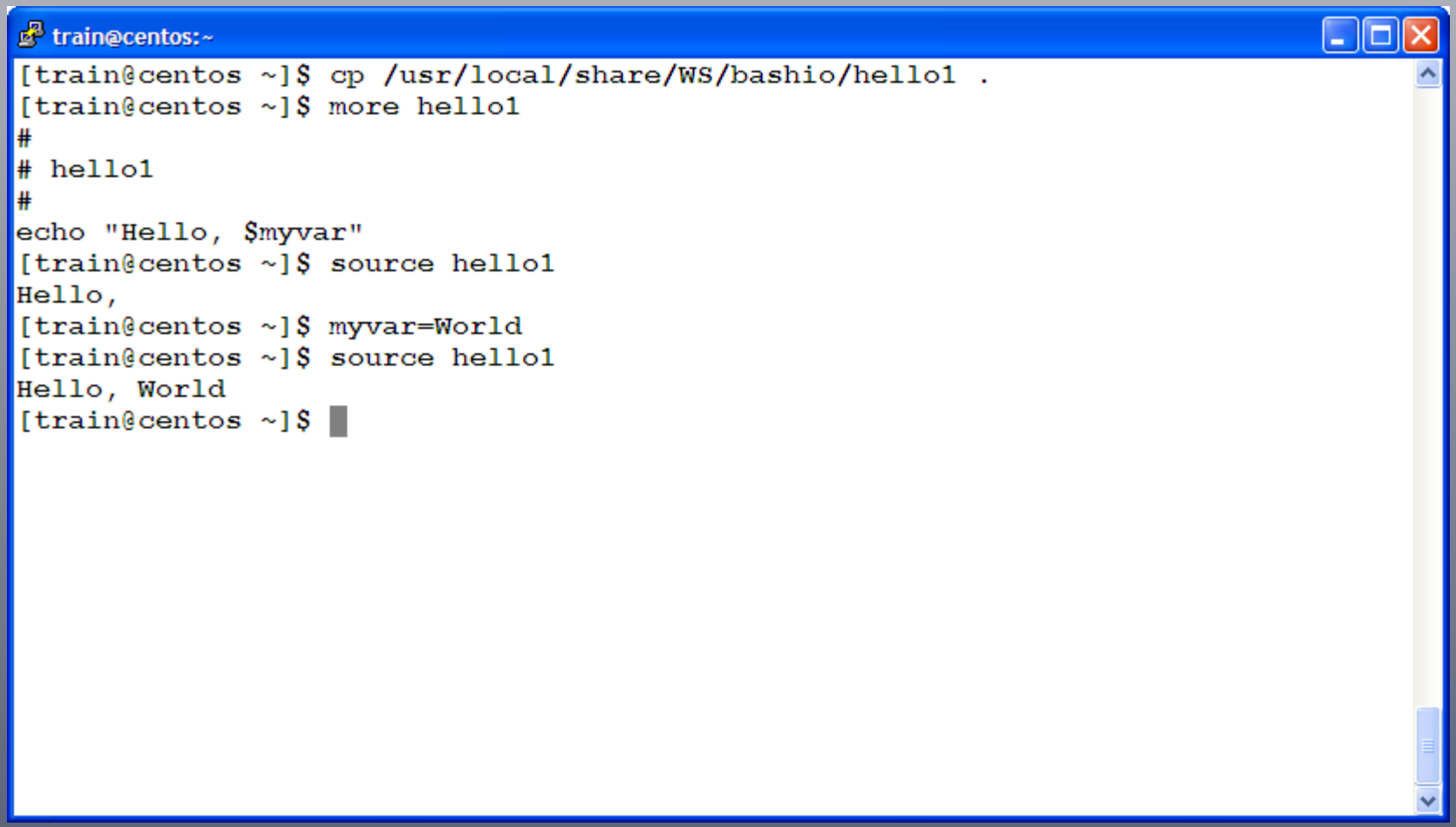


What is a script?

- Nothing more than a list of system commands stored in a file.
- More than just saving time for repetitive tasks.
- Can be modified and customized for particular applications.
- Documents work flow.

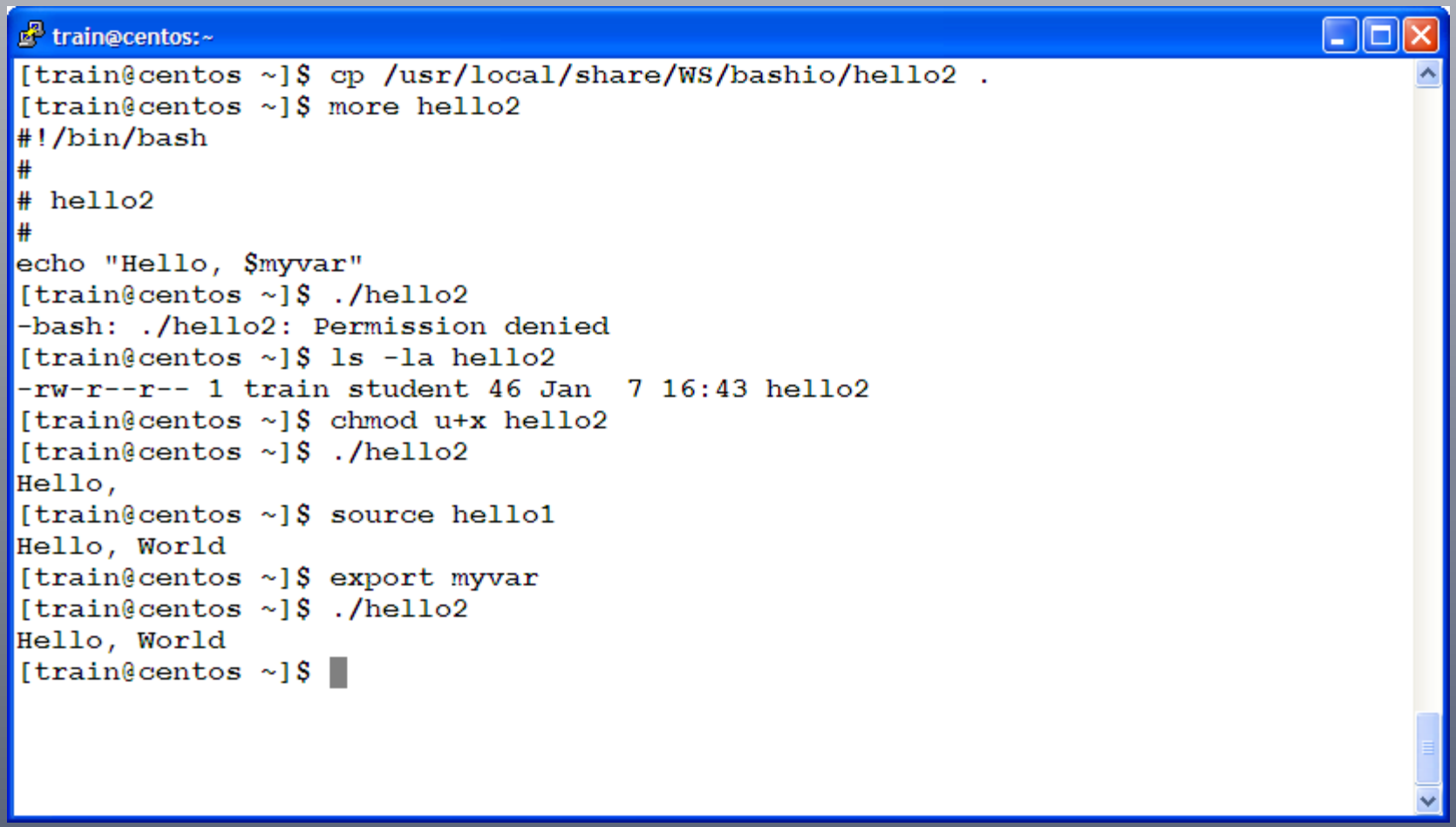
Script Basics: source

- hello1



Script Basics: sha-bang & export

- hello2



Script Basics: Special Characters

- # comment except #! (sha-bang)
- ' ' suppress all meaning (single quotes)
- " " suppress all meaning except $, \, ` (double quotes)
- ` ` value of string is output of the command (back quotes)
- \ to get a literal special character - escape (backslash)
- ; command separator
- spaces are important

Script Basics: Special Characters

- hello3

train@centos:~

```
[train@centos ~]$ cp /usr/local/share/WS/bashio/hello3 .
[train@centos ~]$ more hello3
#!/bin/bash
#
# hello3
#

# Declare myvar
myvar="Hello, World"

# echo variable myvar with single and double quotes
echo 'Single quotes: $myvar'
echo "Double quotes: $myvar"

# show differences with double, single and back quotes
echo "It's \"$myvar\" from the variable \$myvar on: `date`"
[train@centos ~]$ ./hello3
Single quotes: $myvar
Double quotes: Hello, World
It's "Hello, World" from the variable $myvar on: Fri Jan  7 16:46:28 EST 2011
[train@centos ~]$
```

Script Project

Part 1: Build a gnuplot command file (STDOUT).

Part 2: Read a data file (STDIN) and create a new data file suitable for gnuplot using an x, y pair on each line (STDOUT) with error checking (STDERR).

Part 3: Execute the gnuplot command with the command file as the argument.

Part 1: echo

Display message on screen.

echo [*options*]... [*string*]...

-n Do not output the trailing newline.

Part 1: echo

train@centos:~/project

```
[train@centos ~]$ mkdir project
[train@centos ~]$ cd project
[train@centos project]$ cp /usr/local/share/WS/bashio/echo2 .
[train@centos project]$ more echo2
#!/bin/bash

imageFile='fig2.svg'
dataFile='fig2.data'
function_x='0.83037*2.04599**x'

echo -n "\
set terminal svg size 400 300
set output \"$imageFile\"
plot \"$dataFile\" with points pointtype 6, $function_x
"
[train@centos project]$ ./echo2 >commands
[train@centos project]$ wc -l commands
3 commands
[train@centos project]$ more commands
set terminal svg size 400 300
set output "fig2.svg"
plot "fig2.data" with points pointtype 6, 0.83037*2.04599**x
[train@centos project]$
```

Part 1: source & if – then

Run commands from a file.

source *filename* [*arguments*]

Conditionally perform a command.

if [*test-commands*]; **then**
consequent-commands
else
alternate-consequents
fi

Part 1: source & if – then

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/echo3 .
[train@centos project]$ more echo3
#!/bin/bash

source .echorc

if [ "$figTitle" ]; then
  echo -n "\
set title \"$figTitle\"
"
fi

echo -n "\
set terminal svg size $imageWidth $imageHeight
set output \"$imageFile\"
plot \"$dataFile\" with points pointtype 6, $function_x
"
[train@centos project]$
```

Part 1: Testing

Part 1: Testing

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/fig2rc .
[train@centos project]$ more fig2rc
figTitle="figure2: data with function_x 2**x"
imageWidth=500
imageHeight=400
imageFile='fig2.png'
dataFile='fig2.data'
commandFile='fig2commands'
function_x='2**x'
[train@centos project]$ cp fig2rc .echorc
[train@centos project]$ ./echo3
set title "figure2: data with function_x 2**x"
set terminal svg size 500 400
set output "fig2.png"
plot "fig2.data" with points pointtype 6, 2**x
[train@centos project]$ tail -5 fig2rc > .echorc
[train@centos project]$ ./echo3
set terminal svg size  400
set output "fig2.png"
plot "fig2.data" with points pointtype 6, 2**x
[train@centos project]$
```

Part 1: case

Conditionally perform a command.

case *word* **in**
pattern)
command-list
;;
pattern)
command-list
;;
esac

Part 1: case

train@centos:~/project

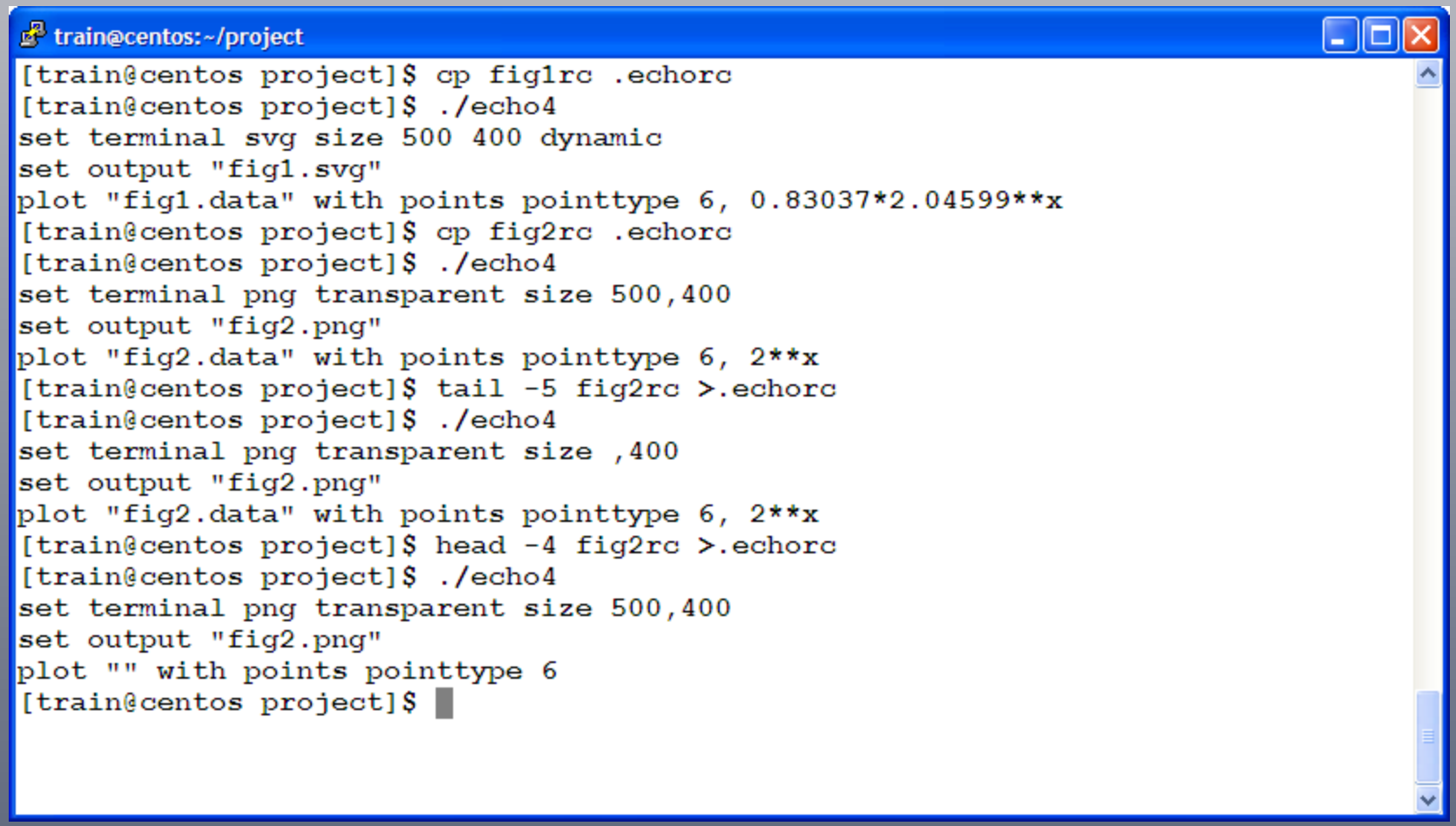
```
[train@centos project]$ cp /usr/local/share/WS/bashio/echo4 .
[train@centos project]$ more echo4
#!/bin/bash

source .echorc

case "$imageFile" in
  *.png )
     echo -n "\
set terminal png transparent size $imageWidth,$imageHeight
set output \"$imageFile\"
"
      ;;
 *.svg)
     echo -n "\
set terminal svg size $imageWidth $imageHeight dynamic
set output \"$imageFile\"
"
esac

echo -n "\
plot \"$dataFile\" with points pointtype 6${function_x:+, }$function_x
"
[train@centos project]$
```

Part 1: Testing



Part 1: function

train@centos:~

```
[train@centos ~]$ cp /usr/local/share/WS/bashio/part1.sh .
[train@centos ~]$ more part1.sh
function gnucommands {
if [ "$figTitle" ]; then
  echo -n "\
set title \"$figTitle\"
"
fi
case "$imageFile" in
  *.png )
     echo -n "\
set terminal png transparent size $imageWidth,$imageHeight
set output \"$imageFile\"
"
      ;;
 *.svg)
     echo -n "\
set terminal svg size $imageWidth $imageHeight dynamic
set output \"$imageFile\"
"
esac
echo -n "\
plot \"$dataFile\" with points pointtype 6${function_x:+, }$function_x
"
}
[train@centos ~]$
```

Part 2: read

Read a line from standard input.

read [-ers] [-a *aname*] [-p *prompt*] [-t *timeout*] [-n *nchars*] [-d *delim*] [*name*...]

-r If this option is given, backslash does not act as an escape character.

Part 2: read

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/read1 .
[train@centos project]$ more read1
#!/bin/bash

read line
echo "$line"
[train@centos project]$ cp /usr/local/share/WS/bashio/read2 .
[train@centos project]$ more read2
#!/bin/bash

read -r x y etc
echo "$x, $y"
[train@centos project]$ ./read1
1 1.8 2 data x y
1 1.8 2 data x y
[train@centos project]$ ./read1
1 1.8\
1 data x y
1 1.81 data x y
[train@centos project]$ ./read2
1 1.8 2 data x y
1, 1.8
[train@centos project]$ ./read2
1 1.8\
1, 1.8\
[train@centos project]$
```

Part 2: if – then – elif

Conditionally perform a command.

if [*test-commands*]; **then**
    consequent-commands
elif [*more-test-commands*]; **then**
    more-consequents
fi

-n True if tests nonzero (contains data).

-z True if tests zero (no data).

Part 2: if – then – elif

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/read3 .
[train@centos project]$ more read3
#!/bin/bash

read -r x y etc
if [ -n "$etc" ]; then
  echo "line too long" >&2
elif [ -z "$y" ]; then
  echo "line too short" >&2
fi
echo $x${y:+, $y}
[train@centos project]$ ./read3
1 1.8
1, 1.8
[train@centos project]$ ./read3
1 1.8 2 data x y
line too long
1, 1.8
[train@centos project]$ ./read3
1
line too short
1
[train@centos project]$
```

Part 2: while

Execute consequent-commands as long as test-commands has an exit status of zero

```
while test-commands; do
    consequent-commands
done
```

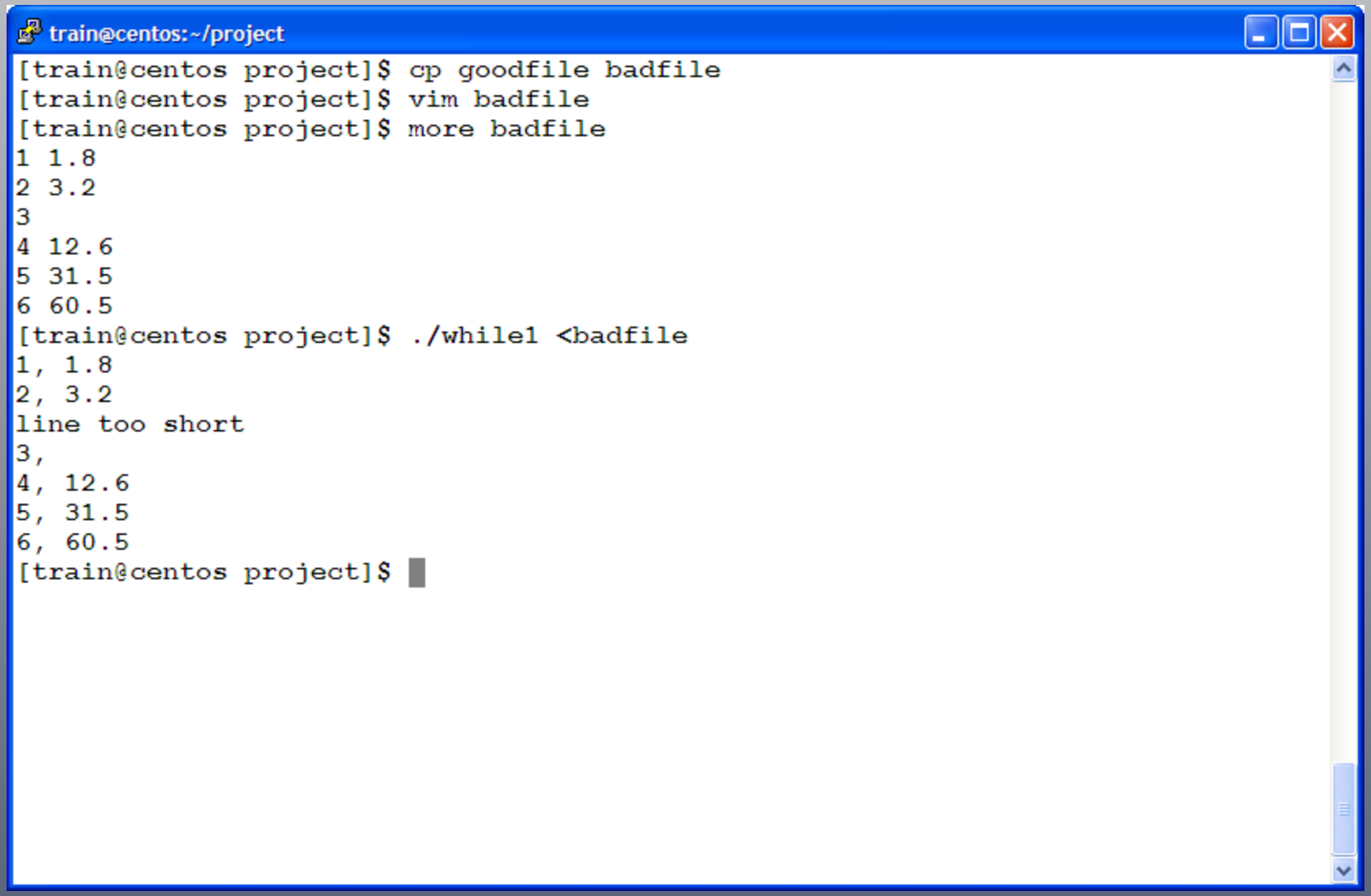
Part 2: while

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/while1 .
[train@centos project]$ more while1
#!/bin/bash

while read -r x y etc; do
  if [ -z "$y" ]; then
    echo "line too short" >&2
  elif [ -n "$etc" ]; then
    echo "line too long, unexpected: $etc" >&2
  fi
  echo "$x, $y"
done
[train@centos project]$ cat > goodfile
1 1.8
2 3.2
3 7.5
4 12.6
5 31.5
6 60.5
[train@centos project]$ ./while1 <goodfile
1, 1.8
2, 3.2
3, 7.5
4, 12.6
5, 31.5
6, 60.5
[train@centos project]$
```

Part 2: Testing



Part 2: Testing

train@centos:~/project

```
[train@centos project]$ cp goodfile warningfile
[train@centos project]$ vim warningfile
[train@centos project]$ more warningfile
1 1.8
2 3.2
3 7.5 4 5
4 12.6
5 31.5
6 60.5 too much data
[train@centos project]$ ./while1 <warningfile
1, 1.8
2, 3.2
line too long, unexpected: 4 5
3, 7.5
4, 12.6
5, 31.5
line too long, unexpected: too much data
6, 60.5
[train@centos project]$
```

Part 2: Testing

train@centos:~/project

```
[train@centos project]$ ./while1 <goodfile >good.data
[train@centos project]$ more good.data
1, 1.8
2, 3.2
3, 7.5
4, 12.6
5, 31.5
6, 60.5
[train@centos project]$ ./while1 <badfile >bad.data
line too short
[train@centos project]$ more bad.data
1, 1.8
2, 3.2
3,
4, 12.6
5, 31.5
6, 60.5
[train@centos project]$ ./while1 <warningfile >warning.data
line too long, unexpected: 4 5
line too long, unexpected: too much data
[train@centos project]$ more warning.data
1, 1.8
2, 3.2
3, 7.5
4, 12.6
5, 31.5
6, 60.5
[train@centos project]$
```

Part 2: let & if

Perform arithmetic on shell variables.

let *expression* [*expression*]

Test-commands using and

if [*expr1* -a *expr2*]; then
if both *expr1* **and** *expr2* are true.
consequent-commands
fi

Part 2: let & if

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/while2 .
[train@centos project]$ more while2
#!/bin/bash

let lineNo=0
while read -r x y etc; do
  let lineNo+=1
  if [ -n "$x" -a -z "$y" ]; then
    echo "line $lineNo too short" >&2
    errCode=1
  elif [ -n "$etc" ]; then
    echo "line $lineNo too long, unexpected $etc" >&2
  fi
  echo $x${y:+, $y}
done
[ -z "$errCode" ]
[train@centos project]$ ./while2 <goodfile >good.data && echo "good data file"
good data file
[train@centos project]$ ./while2 <badfile >bad.data && echo "good data file"
line 3 too short
[train@centos project]$ ./while2 <warningfile > warning.data && echo "good data
file"
line 3 too long, unexpected 4 5
line 6 too long, unexpected too much data
good data file
[train@centos project]$
```

Part 2: return

Cause a shell function to exit with the return value n.

return [n]

Part 2: function

train@centos:~

```
[train@centos ~]$ cp /usr/local/share/WS/bashio/part2.sh .
[train@centos ~]$ more part2.sh
function datafile {
let returncode=0
let lineNo=0
while read -r x y etc; do
  let lineNo+=1
  if [ -n "$x" -a -z "$y" ]; then
    echo "line $lineNo too short" >&2
    returncode=1
  elif [ -n "$etc" ]; then
    echo "line $lineNo too long, unexpected $etc" >&2
  fi
  echo $x${y:+, $y}
done
return $returncode
}
[train@centos ~]$
```

Part 3: Putting it all together

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/makefig1 .
[train@centos project]$ more makefig1
#!/bin/bash
# makefig
#    reads data file and makes a gnuplot figure
#
# Get functions:
#    die, gnucommands, dataFile
source functions.sh
#-----
# Get variables from run control file:
#    dataFile, commandFile, imageFile, imageHeight, imageWidth, function_x
[ -e .makefigrc ] || die "file \".makefigrc\" does not exist"
source .makefigrc
#-----
[ "$dataFile" ] || die "no data file"
commandFile=${commandFile:-$dataFile.gnuplot}
#-----
# Make output files:
#   dataFile, commandFile, imageFile
datafile >$dataFile || die "some lines to short"
gnucommands >$commandFile
gnuplot $commandFile
[train@centos project]$
```

Part 3: functions.sh

train@centos:~

```
[train@centos ~]$ cp /usr/local/share/WS/bashio/functions.sh .
[train@centos ~]$ more functions.sh
# Define functions:
#    die, gnucommands, dataFile
function die {
  echo "makefig: $@" >&2
  exit 1
}
function gnucommands {
  if [ "$figTitle" ]; then
    echo -n "\
set title \"$figTitle\"
"
  fi
  case "$imageFile" in
    *.png )
      echo -n "\
set terminal png transparent size $imageWidth,$imageHeight
set output \"$imageFile\"
"
      ;;
    *.svg )
      echo -n "\
set terminal svg size $imageWidth $imageHeight dynamic
set output \"$imageFile\"
"
  esac
--More--(52%)
```

Part 3: Testing

train@centos:~/project

```
[train@centos project]$ cp fig1rc .makefigrc
[train@centos project]$ ./makefig1 <badfile && echo "figure ready"
line 3 too short
makefig: some lines to short
[train@centos project]$ ./makefig1 <warningfile && echo "figure ready"
line 3 too long, unexpected 4 5
line 6 too long, unexpected too much data
figure ready
[train@centos project]$ alias makefig=./makefig1
[train@centos project]$ alias
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias makefig='./makefig1'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
[train@centos project]$ makefig <goodfile && echo "figure ready"
figure ready
[train@centos project]$ firefox fig1.svg
[train@centos project]$
```

Part 3: Testing

train@centos:~/project

```
[train@centos project]$ firefox

[1]+  Stopped                 firefox
[train@centos project]$ bg
[1]+ firefox &
[train@centos project]$ jobs
[1]+  Running                 firefox &
[train@centos project]$
[1]+  Done                    firefox
[train@centos project]$ jobs
[train@centos project]$ firefox &
[1] 16636
[train@centos project]$ jobs
[1]+  Running                 firefox &
[train@centos project]$ cp fig2rc .makefigrc
[train@centos project]$ makefig <goodfile && echo "figure ready"
figure ready
[train@centos project]$ firefox fig2.png
[train@centos project]$ makefig <warningfile && echo "figure ready"
line 3 too long, unexpected 4 5
line 6 too long, unexpected too much data
figure ready
[train@centos project]$ firefox fig2.png
[train@centos project]$
```

Part 3: command line options

train@centos:~/project

```
[train@centos project]$ cp /usr/local/share/WS/bashio/makefig2 .
[train@centos project]$ more makefig2
#!/bin/bash
# makefig:
#    takes std input data file and makes a gnuplot figure
# options:
#    -v                for more reporting
#    -f filename       to set run control file
# arguments:
#    functions of x to be added to the figure
#-----
# Get functions:
#    die, gnucommands, dataFile
source functions.sh
#-----
# Get variables for argument list
#    verbose  0|1
#    rcfile   run control file with assignments
#    argfuns  list of functions to plot
rcfile='.makefigrc'
verbose=0
while [ $# -gt 0 ]; do
  case $1 in
    -v)
       verbose=1
       ;;
    -f)
       shift
```

Part 3: command line options

train@centos:~/project

```
        rcfile="$1"
        ;;
    -*)
        die "illegal option $1
Usage: `basename $0` [-v] [-f file] [function ...]"
        ;;
     *)
        argfuns="$argfuns${argfuns:+, }$1"
  esac
  shift
done
[ -e "$rcfile" ] || die "file \"$rcfile\" does not exist"
#-----
# Get variables for run control file
#    figTitle, imageWidth, imageHeight, imageFile, dataFile
source $rcfile
[ "$dataFile" ] || die "no data file name"
[ "$imageFile" ] || die "no image file name"
[ "$imageHeight" -a "$imageWidth" ] || die "no plot dimensions"
#-----
# Make the file
#     dataFile
datafile >|$dataFile || die "some lines too short"
function_x="$function_x${argfuns:+, }$argfuns"
#-----
# print formated report
--More--(79%)
```

Part 3: command line options

train@centos:~/project

```
[ -e "$rcfile" ] || die "file \"$rcfile\" does not exist"
#-----
# Get variables for run control file
#    figTitle, imageWidth, imageHeight, imageFile, dataFile
source $rcfile
[ "$dataFile" ] || die "no data file name"
[ "$imageFile" ] || die "no image file name"
[ "$imageHeight" -a "$imageWidth" ] || die "no plot dimensions"
#-----
# Make the file
#     dataFile
datafile >|$dataFile || die "some lines too short"
function_x="$function_x${argfuns:+, }$argfuns"
#-----
# print formated report
withFun=${function_x:+, together with function $function_x}
[ $verbose -eq 0 ] || echo "
${figTitle:-figure:}
   Make a plot of data points from the file
   $dataFile$withFun.
   The plot will be sized at $imageWidth by $imageHeight, and stored
   in the file $imageFile.
" | fmt
#-----
# Make figure
gnucommands | gnuplot
[train@centos project]$
```

Part 3: Testing

train@centos:~/project

```
[train@centos project]$ alias makefig=./makefig2
[train@centos project]$ more fig1rc
imageWidth=500
imageHeight=400
imageFile='fig1.svg'
dataFile='fig1.data'
commandFile='fig1commands'
function_x='0.83037*2.04599**x'
figTitle='data with fitted exponential'
[train@centos project]$ sed 's/.svg/.png/' fig1rc > .makefigrc
[train@centos project]$ more .makefigrc
imageWidth=500
imageHeight=400
imageFile='fig1.png'
dataFile='fig1.data'
commandFile='fig1commands'
function_x='0.83037*2.04599**x'
figTitle='data with fitted exponential'
[train@centos project]$ makefig -h <goodfile && firefox fig1.png
makefig: illegal option -h
Usage: makefig2 [-v] [-f file] [function ...]
[train@centos project]$ makefig -v <goodfile && firefox fig1.png

data with fitted exponential
   Make a plot of data points from the file fig1.data, together with
   function 0.83037*2.04599**x.  The plot will be sized at 500 by 400,
   and stored in the file fig1.png.
```

Part 3: Testing

train@centos:~/project

```
[train@centos project]$ makefig -v -f fig2rc "0.83037*2.04599**x" <badfile && fi
refox fig2.png
line 3 too short
makefig: some lines too short
[train@centos project]$ makefig -v -f fig2rc "0.83037*2.04599**x" <goodfile && f
irefox fig2.png

figure2: data with function_x 2**x
   Make a plot of data points from the file fig2.data, together with
   function 2**x, 0.83037*2.04599**x.  The plot will be sized at 500 by
   400, and stored in the file fig2.png.

[train@centos project]$ cp /usr/local/share/WS/bashio/fig3rc .
[train@centos project]$ more fig3rc
imageWidth=500
imageHeight=400
imageFile="fig3.png"
dataFile="fig3.data"
commandFile="fig3commands"
figTitle="figure3: nearly exponential data"
[train@centos project]$ makefig -v -f fig3rc <goodfile && firefox fig3.png

figure3: nearly exponential data
   Make a plot of data points from the file fig3.data.  The plot will
   be sized at 500 by 400, and stored in the file fig3.png.

[train@centos project]$
```

Resources

- Bash scripting Tutorial
  http://www.linuxconfig.org/Bash_scripting_Tutorial
- Advanced Bash-Scripting Guide
  http://tldp.org/LDP/abs/html/
- VTC (Unix Shell Scripting Advanced) – need to request an account
  http://www.udel.edu/it/learnit/course/vtccom.html