```
#!/bin/more
==============================
SCRIPT NAME: arrayex.sh
==============================
#!/bin/bash
# simple array list and loop for display

SERVERLIST=("websrv01" "websrv02" "websrv03" "websrv04")
COUNT=0

for INDEX in ${SERVERLIST[@]}; do
  echo "Processing Server: ${SERVERLIST[COUNT]}"
  COUNT="`expr $COUNT + 1`"
done
==============================
SCRIPT NAME: casesample.sh
==============================
#!/bin/bash
# demo of the case statement

clear

echo "MAIN MENU"
echo "========="
echo "1) Choice One"
echo "2) Choice Two"
echo "3) Choice Three"
echo ""
echo "Enter Choice: "
read MENUCHOICE

case $MENUCHOICE in
  1)
    echo "Congratulations for Choosing the First Option";;
  2)
    echo "Choice 2 Chosen";;
  3)
    echo "Last Choice Made";;
  *)
    echo "You chose unwisely";;
esac
==============================
SCRIPT NAME: checkargs2.sh
==============================
#!/bin/bash

: ${3?"USAGE: $1 ARGUMENT $2 ARGUMENT $3 ARGUMENT"}

echo "I got all three!"
==============================
SCRIPT NAME: checkargs.sh
==============================
#!/bin/bash
```

CH7.7

CH7.6

```
if [ "$#" != "3" ]; then
  echo "USAGE: checkargs.sh [parm1] [parm2] [parm3]"
  exit 300
fi

echo "I live! I got what I needed!"
==============================
SCRIPT NAME: cmdlinevar.sh
==============================
#!/bin/bash
# demo of command line values passed in with our shell script

USERNAME=$1
PASSWORD=$2
echo "The following Username is $USERNAME and Password is $PASSWORD"
==============================
SCRIPT NAME: comments.sh
==============================
#!/bin/bash
# This line is intended to be used as a general description of the script
# and anything that it does

clear    # clears the screen

# MYUSERNAME="Terry"    # the username for this application
MYUSERNAME="Don" # new username added later

echo "We are using the default user called: $MYUSERNAME" # display to the
console

DATETIMESTAMP=`date`

echo "This is when the script was run: $DATETIMESTAMP" # this is the
timestamp of run
==============================
SCRIPT NAME: env.sh
==============================
#!/bin/bash

clear

echo "This script will give us environment information"
echo "================================================"
echo ""
echo "Hello Username: $USER"
echo ""
echo "Your Home Directory is: $HOME"
echo ""
echo "Your History File Will Ignore: $HISTCONTROL"
echo ""
echo "Your Terminal Session Type is: $TERM"
echo ""
==============================
SCRIPT NAME: errorexit.sh
```

```
==============================
#!/bin/bash
# demo of using error handling with exit

echo "Change to a directory and list the contents"
DIRECTORY=$1

cd $DIRECTORY 2>/dev/null

if [ "$?" = "0" ]; then
  echo "We can change into the directory $DIRECTORY, and here are the
contents"
  echo "`ls -al`"
else
  echo "Cannot change directories, exiting with an error and no listing"
  exit 111
fi
==============================
SCRIPT NAME: errors.sh
==============================
#!/bin/bash
# this is to show exit status types
set -e

expr 1 + 5
echo $?

rm doodles.sh
echo $?

expr 10 + 10
echo $?
==============================
SCRIPT NAME: execops.sh
==============================
#!/bin/bash
# execution operators example

echo "Enter a number between 1 and 5: "
read VALUE

if [ "$VALUE" -eq "1" ] || [ "$VALUE" -eq "3" ] || [ "$VALUE" -eq "5" ];
then
  echo "You entered the ODD value of $VALUE"
else
  echo "You entered a value of $VALUE"
fi
==============================
SCRIPT NAME: expressions.sh
==============================
#!/bin/bash
# expression evaluation

expr 2 + 2
```

```
expr 2 + 2 \* 4

expr \( 2 + 2 \) \* 4
==============================
SCRIPT NAME: ex.sh
==============================
#!/bin/bash

FINDUSER=`find /home -user user`
alias finduser="find /home -user user"

echo "Variable: $FINDUSER"
VARFIND=`finduser`
echo "ALIAS: $VARFIND"
==============================
SCRIPT NAME: filedesc.sh
==============================
#!/bin/bash
# demo of reading and writing to a file using a file descriptor

echo "Enter a file name to read: "
read FILE

exec 5<>$FILE

while read -r SUPERHERO; do
  echo "Superhero Name: $SUPERHERO"
done <&5

echo "File Was Read On: `date`" >&5

exec 5>&-
==============================
SCRIPT NAME: forsample.sh
==============================
#!/bin/bash
# this is a demo of the for loop

echo "List all the shell scripts contents of the directory"

SHELLSCRIPTS=`ls *.sh`

for SCRIPT in "$SHELLSCRIPTS"; do
  DISPLAY="`cat $SCRIPT`"
  echo "File: $SCRIPT - Contents $DISPLAY"
done
==============================
SCRIPT NAME: funcparms.sh
==============================
#!/bin/bash
# this demo is for functional parameter passing

# global variable
```

```
USERNAME=$1

# function definitions - start

# calculate age in days
funcAgeInDays () {
  echo "Hello $USERNAME, You are $1 Years Old."
  echo "That makes you approximately `expr $1 \* 365` days old..."
}

# function definitions - stop

# scrip - start
clear

echo "Enter Your Age: "
read USERAGE

# calculate the number of days
funcAgeInDays $USERAGE
==============================
SCRIPT NAME: funcstruct.sh
==============================
#!/bin/bash
# demo of functions within a shell script structure

# script or global variables
CMDLINE=$1

# function definitions - start

# displays a message
funcExample () {
  echo "This is an example"
}

# display another message
funcExample2 () {
  echo "This is another example"
}

# function definitions - stop

# beginning of the script
echo "This is the start..."

funcExample2
funcExample
funcExample
==============================
SCRIPT NAME: ifexpr.sh
==============================
#!/bin/bash
# test multiple expressions in single if statement
```

```
FILENAME=$1

echo "Testing for file $FILENAME and readability"

if [ -f $FILENAME ] && [ -r $FILENAME ]
  then
    echo "File $FILENAME exists AND is readable"
fi
==============================
SCRIPT NAME: ifsdelim.sh
==============================
#!/bin/bash
# delimiter example using IFS

echo "Enter filename to parse: "
read FILE

echo "Enter the Delimiter: "
read DELIM

IFS="$DELIM"

while read -r CPU MEMORY DISK; do
  echo "CPU: $CPU"
  echo "Memory: $MEMORY"
  echo "Disk: $DISK"
done <"$FILE"
==============================
SCRIPT NAME: ifthenelse.sh
==============================
#!/bin/bash
# simple example of if then else and nested if statements

clear

echo "Enter a number between 1 and 3:"
read VALUE

if [ "$VALUE" -eq "1" ] 2>/dev/null; then
  echo "You entered #1"
elif [ "$VALUE" -eq "2" ] 2>/dev/null; then
  echo "You successfully entered #2"
elif [ "$VALUE" -eq "3" ] 2>/dev/null; then
  echo "You entered the 3rd number"
else
  echo "You didn't follow the directions!"
fi
==============================
SCRIPT NAME: makedoc.sh
==============================
#!/bin/bash

DOCFILE="script_listing"
```

CH7.8

```
echo "#!/bin/more" > "$DOCFILE"

ls *.sh > tmplisting.txt

while IFS= read -r FILENAME; do
  if [ -f "$FILENAME" ]; then
    echo "==============================" >> "$DOCFILE"
    echo "SCRIPT NAME: $FILENAME " >> "$DOCFILE"
    echo "==============================" >> "$DOCFILE"
    echo ""
    echo "`cat $FILENAME`" >> "$DOCFILE"
  fi
done < tmplisting.txt

chmod 755 "$DOCFILE"

rm tmplisting.txt
==============================
SCRIPT NAME: nested.sh
==============================
#!/bin/bash
# demo of nested functions and some abstraction

# global variable
GENDER=$1

# function definitions - start

# create a human being
funcHuman () {
  ARMS=2
  LEGS=2

  echo "A Human has $ARMS arms and $LEGS legs - but what gender are we?"
  echo ""

  funcMale () {
    BEARD=1

    echo "This man has $ARMS arms and $LEGS legs, with $BEARD
beard(s)..."
    echo ""
  }

  funcFemale () {
    BEARD=0

    echo "This woman has $ARMS arms and $LEGS legs, with $BEARD
beard(s)..."
    echo ""
  }
}
```

```
# function definitions - stop

# script - start
clear
echo "Determining the characteristics of the gender $GENDER"
echo ""

# determine the actual gender and display the characteristics
if [ "$GENDER" == "male" ]; then
  funcHuman
  funcMale
else
  funcHuman
  funcFemale
fi
==============================
SCRIPT NAME: null.sh
==============================
#!/bin/bash
# redirect to /dev/null example

echo "This is displaying on the console"

echo "This is going into the black hole" >> /dev/null
==============================
SCRIPT NAME: override2.sh
==============================
#!/bin/bash
# override/trap the system exit and execute a custom function

# global variables
TMPFILE="tmpfile.txt"
TMPFILE2="tmpfile2.txt"

trap 'funcMyExit' EXIT

# function declarations - start

# run this exit instead of the default exit when called
funcMyExit () {
  echo "Exit Intercepted..."
  echo "Cleaning up the temp files..."
  rm -rf "tmpfil*.txt"
  exit 255
}

# function declarations - stop

# script - start

echo "Write something to tmp file for later use..." > $TMPFILE
echo "Write something to tmp file 2 for later user..." > $TMPFILE2

echo "Trying to copy the indicated file before processing..."
```

```
cp -rf $1 newfile.txt 2>/dev/null

if [ "$?" -eq "0" ]; then
  echo "Everything worked out ok..."
else
  echo "I guess it did not work out ok..."
  exit 1
fi

# script - stop
==============================
SCRIPT NAME: overriding.sh
==============================
#!/bin/bash
# override/trap the system exit and execute a custom function

# global variables
TMPFILE="tmpfile.txt"
TMPFILE2="tmpfile2.txt"

trap 'funcMyExit' EXIT

# function declarations - start

# run this exit instead of the default exit when called
funcMyExit () {
  echo "Exit Intercepted..."
  echo "Cleaning up the temp files..."
  rm -rf tmpfil*.txt
  exit 255
}

# function declarations - stop

# script - start

echo "Write something to tmp file for later use..." > $TMPFILE
echo "Write something to tmp file 2 for later user..." > $TMPFILE2

echo "Trying to copy the indicated file before processing..."
cp -rf $1 newfile.txt 2>/dev/null

if [ "$?" -eq "0" ]; then
  echo "Everything worked out ok..."
else
  echo "I guess it did not work out ok..."
  exit 1
fi

# script - stop
==============================
SCRIPT NAME: readfile.sh
==============================
#!/bin/bash
```

CH7.5

```
# simple file reading (non-binary) and displaying one line at a time

echo "Enter a filename to read: "
read FILE

while read -r SUPERHERO; do
  echo "Superhero Name: $SUPERHERO"
done < "$FILE"
==============================
SCRIPT NAME: readsample.sh
==============================
#!/bin/bash
# interactive script for user input

echo "Enter Your First Name: "
read FIRSTNAME
echo "Enter Your Last Name: "
read LASTNAME

echo ""
echo "Your Full Name is: $FIRSTNAME $LASTNAME"
echo ""
echo "Enter Your Age: "
read USERAGE

echo "In 10 Years, You will be `expr $USERAGE + 10` years old."
==============================
SCRIPT NAME: returnval.sh
==============================
#!/bin/bash
# demo of return values and testing results

# global variable
YES=0
NO=1
FIRST=$1
SECOND=$2
THIRD=$3

# function definitions - start

# check the command line parameters passed in
funcCheckParms () {
  # did we get three
  if [ ! -z "$THIRD" ]; then
    echo "We got three parms..."
    return $YES
  else
    echo "We did not get three parms..."
    return $NO
  fi
}

# function definitions - stop
```

```
# script - start
funcCheckParms
RETURN_VALS=$?

# did we get three or not?
if [ "$RETURN_VALS" -eq "$YES" ]; then
  echo "We received three parms and they are: "
  echo "Parm 1: $FIRST"
  echo "Parm 2: $SECOND"
  echo "Parm 3: $THIRD"
  echo ""
else
  echo "Usage: returnval.sh [parm1] [parm2] [parm3]"
  exit 1
fi
==============================
SCRIPT NAME: simpledialog.sh
==============================
#!/bin/bash
# demo of a dialog box that will display a menu

# global variables / default values
MENUBOX=${MENUBOX=dialog}

# function declarations - start

# function to display a simple menu
funcDisplayDialogMenu () {
  $MENUBOX --title "[ M A I N   M E N U ]" --menu "Use UP/DOWN Arrows to
Move and Select or the Number of Your Choice and Enter" 15 45 4 1
"Display Hello World" 2 "Display Goodbye World" 3 "Display Nothing" X
"Exit" 2>choice.txt
}

# function declarations - stop

# script - start

funcDisplayDialogMenu

case "`cat choice.txt`" in
  1) echo "Hello World";;
  2) echo "Goodbye World";;
  3) echo "Nothing";;
  X) echo "Exit";;
esac

# script - stop
==============================
SCRIPT NAME: simplefunc.sh
==============================
#!/bin/bash
# this is a simple function example
```

```
echo "Starting the function definition..."

funcExample () {
echo "We are now INSIDE the function..."
}


funcExample
==============================
SCRIPT NAME: simpleif.sh
==============================
#!/bin/bash
# simple if script for guessing a number

echo "Guess the Secret Number"
echo "======================="
echo ""
echo "Enter a Number Between 1 and 5: "
read GUESS

if [ $GUESS -eq 3 ]
  then
    echo "You Guessed the Correct Number!"
fi
==============================
SCRIPT NAME: simpleinfobox.sh
==============================
#!/bin/bash
# demo of a simple info box with dialog and ncurses

# global variables / default values
INFOBOX=${INFOBOX=dialog}
TITLE="Default"
MESSAGE="Something to say"
XCOORD=10
YCOORD=20

# function declarations - start

# display the infobox and our mesage
funcDisplayInfoBox () {
  $INFOBOX --title "$1" --infobox "$2" "$3" "$4"
  sleep "$5"
}

# function declarations - stop

# script - start

if [ "$1" == "shutdown" ]; then
  funcDisplayInfoBox "WARNING!" "We are SHUTTING DOWN the System..." "11"
"21" "5"
  echo "Shutting Down!"
```

CH7.1

```
else
  funcDisplayInfoBox "Information..." "You are not doing anything fun..."
"11" "21" "5"
  echo "Not doing anything..."
fi

# script - stop
==============================
SCRIPT NAME: simpleinputbox.sh
==============================
#!/bin/bash
# simple demo of an input dialog box

# global variables / default values
INPUTBOX=${INPUTBOX=dialog}
TITLE="Default"
MESSAGE="Something to display"
XCOORD=10
YCOORD=20

# function declarations - start

# display the input box
funcDisplayInputBox () {
  $INPUTBOX --title "$1" --inputbox "$2" "$3" "$4" 2>tmpfile.txt
}

# function declarations - stop

# script - start

funcDisplayInputBox "Display File Name" "Which file in the current
directory do you want to display?" "10" "20"

if [ "`cat tmpfile.txt`" != "" ]; then
  cat "`cat tmpfile.txt`"
else
  echo "Nothing to do"
fi

# script - stop
==============================
SCRIPT NAME: simplemsgbox.sh   CH7.2
==============================
#!/bin/bash
# demo of a message box with an OK button

# global variables / default variables
MSGBOX=${MSGBOX=dialog}
TITLE="Default"
MESSAGE="Some Message"
XCOORD=10
YCOORD=20
```

```
# function declarations - start

# display the message box with our message
funcDisplayMsgBox () {
  $MSGBOX --title "$1" --msgbox "$2" "$3" "$4"
}

# function declarations - stop

# script - start
if [ "$1" == "shutdown" ]; then
  funcDisplayMsgBox "WARNING!" "Please press OK when you are ready to
shut down the system" "10" "20"
  echo "SHUTTING DOWN NOW!!!"
else
  funcDisplayMsgBox "Boring..." "You are not asking for anything fun..."
"10" "20"
  echo "Not doing anything, back to regular scripting..."
fi

# script - stop
==============================
SCRIPT NAME: substitution.sh
==============================
#!/bin/bash
# This script is intended to show how to do simple substitution

shopt -s expand_aliases

alias TODAY="date"
alias UFILES="find /home -user user"

TODAYSDATE=`date`
USERFILES=`find /home -user user`

echo "Today's Date: $TODAYSDATE"
echo "All Files Owned by USER: $USERFILES"

A=`TODAY`
B=`UFILES`

echo "With Alias, TODAY is: $A"
echo "With Alias, UFILES is: $B"
==============================
SCRIPT NAME: testfile.sh
==============================
#!/bin/bash
# tests for existence of indicated file name

FILENAME=$1
echo "Testing for the Existence of a File called $FILENAME"

if [ ! -f $FILENAME ]
  then
```

```
    echo "File $FILENAME Does NOT Exist!"
fi
==============================
SCRIPT NAME: test.sh
==============================
#!/bin/bash

clear

echo "Hello World"
==============================
SCRIPT NAME: trapex.sh
==============================
#!/bin/bash
# example of trapping events and limiting the shell stopping

clear

trap 'echo " - Please Press Q to Exit.."' SIGINT SIGTERM SIGTSTP

while [ "$CHOICE" != "Q" ] && [ "$CHOICE" != "q" ]; do
  echo "MAIN MENU"
  echo "========="
  echo "1) Choice One"
  echo "2) Choice Two"
  echo "3) Choice Three"
  echo "Q) Quit/Exit"
  echo ""
  read CHOICE

  clear
done
==============================
SCRIPT NAME: varexample.sh
==============================
#!/bin/bash

MYUSERNAME="username"
MYPASSWORD="password123"
STARTOFSCRIPT=`date`

echo "My login name for this application is: $MYUSERNAME"
echo "My login password for this application is: $MYPASSWORD"
echo "I started this script at: $STARTOFSCRIPT"

ENDOFSCRIPT=`date`

echo "I ended this script at: $ENDOFSCRIPT"
==============================
SCRIPT NAME: varscope.sh
==============================
#!/bin/bash
# demonstrating variable scope
```

```
# global variable declaration
GLOBALVAR="Globally Visible"

# function definitions - start

# sample function for function variable scope
funcExample () {
  # local variable to the function
  LOCALVAR="Locally Visible"

  echo "From within the function, the variable is $LOCALVAR..."
}

# functions definitions - stop

# script - start
clear

echo "This step happens first..."
echo ""
echo "GLOBAL variable = $GLOBALVAR (before the function call)"
echo "LOCALVAR variable = $LOCALVAR (before the function call)"
echo ""
echo "Calling Function - funcExample()"
echo ""

funcExample

echo ""
echo "Function has been called..."
echo ""
echo "GLOBAL variable = $GLOBALVAR (after the function call)"
echo "LOCALVAR variable = $LOCALVAR (after the function call)"
==============================
SCRIPT NAME: whilesample.sh
==============================
#!/bin/bash
# while loop example

echo "Enter the number of times to display the 'Hello World' message"
read DISPLAYNUMBER

COUNT=1

while [ $COUNT -le $DISPLAYNUMBER ]
do
  echo "Hello World - $COUNT"
  COUNT="`expr $COUNT + 1`"
done
```