Shell assignment from question 1 - 28

------------------------------------------------------------------------------------------------------------------------------------------

1->

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 ""

2)

#!/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"

3)

#!/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

4)

#!/bin/bash

# redirect to /dev/null example

echo "This is displaying on the console"

echo "This is going into the black hole" >> /dev/null

5)

#!/bin/bash

# this is to show exit status types

set -e

expr 1 + 5

echo $?

rm doodles.sh

echo $?

expr 10 + 10

echo $?

6)

#!/bin/bash

# expression evaluation

expr 2 + 2

expr 2 + 2 \\* 4

expr \( 2 + 2 \) \\* 4

7)

#!/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"

8)

#!/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."

9)

#!/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

10)

#!/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"

11)

#!/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

12)

#!/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

13)

#!/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

14)

#!/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

15)

#!/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

16)

#!/bin/bash

# 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"

17)

#!/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>&-

18)

#!/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"

19)

#!/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

20)

#!/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

21)

**#!/bin/bash**

**# this is a simple function example**

**echo "Starting the function definition..."**

**funcExample () {**

**echo "We are now INSIDE the function..."**

**}**

**echo "But we did NOT call the function, yet..."**

**echo "NOW let's call it"**

**# call the function**

**funcExample**

**echo "...and back outside the function, continuing to the next command."**

**echo "That's it"**

**22)**

#!/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)"

23)

#!/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

24)

#!/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

25)

#!/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!"

else

funcDisplayInfoBox "Information..." "You are not doing anything fun..." "11" "21" "5"

echo "Not doing anything..."

fi

# script – stop

26)

#!/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

27)

#!/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

28)

#!/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

-----------------------------------------------------------------------------------------------------