Shell scripting

1 Create a script that, when run, will display the following environment variables to the console:

USER

HOME

HISTCONTROL

TERM

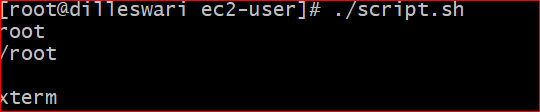
#! /bin/bash

Printenv USER

Printenv Home

Printenv TERM

Printenv HISTROCONTROL



2 Write a script that sets FOUR variables:

MYUSERNAME

MYPASSWORD

STARTOFSCRIPT

ENDOFSCRIPT

Populate the first two with some default value and use command redirection to set the third and fourth value to the date/time of when the script was started and completed. Within the script, be sure to display the values to the terminal when run.

#!/bin/sh

MYUSERNAME=$USER

MYPASSWORD='password'

STARTOFSCRIPT=`date`

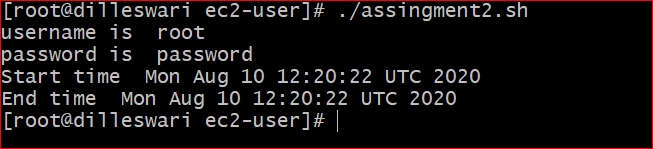
ENDOFSCRIPT=`date`

echo 'username is ' $MYUSERNAME

echo 'password is ' $MYPASSWORD

echo 'Start time ' $STARTOFSCRIPT

echo 'End time ' $ENDOFSCRIPT



3

Develop a script that creates, sets and displays two variables to the terminal when run. Within this script, add comments to explain what the script is doing, what each variable is and, using inline comments, what each command is doing.

#!/bin/bash

# This script clears the terminal, displays a greeting and gives information

# about currently connected users. The two example variables are set and displayed.

clear # clear terminal window

echo "The script starts now."

echo "Hi, $USER!" # dollar sign is used to get content of variable

echo

echo "I will now fetch you a list of connected users:"

echo

w # show who is logged on and

echo # what they are doing

echo "I'm setting two variables now."

MYUSERNAME="laxmi" # set a local shell variable

MYPASSWORD="passwd" # set a local shell variable

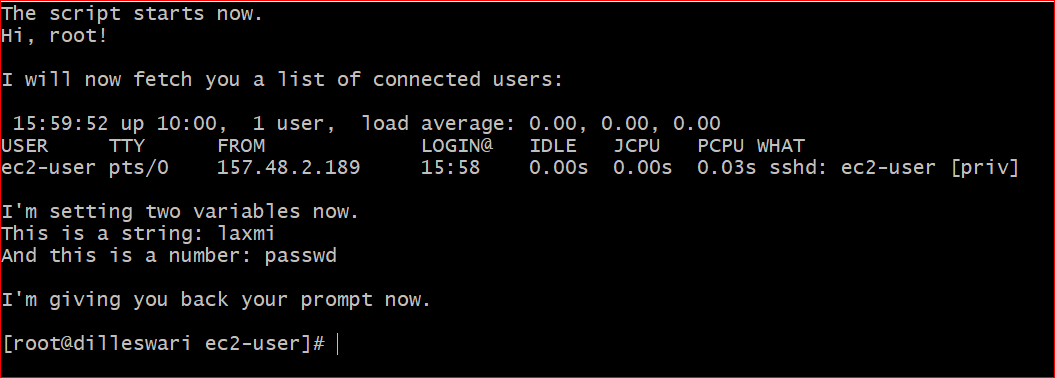
echo "This is a string: $MYUSERNAME" # display content of variable

echo "And this is a number: $MYPASSWORD" # display content of variable

echo

echo "I'm giving you back your prompt now."

echo



4 Create a simple script that does the following:

Echo a full sentence to the terminal

Echo a different full sentence, but redirect it to /dev/null

Run and display the results and make sure the statements appear where intended.

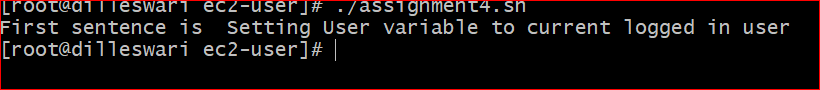
#! /bin/bash

firstsentence='Setting User variable to current logged in user'

secendsentence='Setting pwd to present working directory'

echo 'First sentence is ' $firstsentence

echo 'Second sentence is '$secondsentence >> /dev/null



5 write a script that runs three commands:

Evaluate an arithmetic expression

Attempt to remove a file that does not exist in the current directory

Evaluate another arithmetic expression

Immediately after each command, echo the exit status of that command to the terminal using the appropriate variable to show success and failure exit codes.

#!/bin/bash

a=10

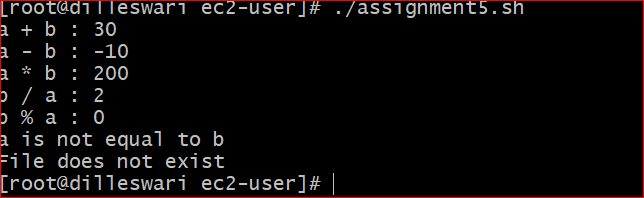
b=20

val=`expr $a + $b`

echo "a + b : $val"

val=`expr $a - $b`

echo "a - b : $val"



6 Write a script that evaluates and displays the following arithmetic operations:

Add two numbers

Add two numbers and multiply by a third, do not group anything

Add two numbers, grouped (changing order of precedence) and multiply by a third

Keep in mind special characters and/or escape characters as needed.

#! /bin/bash

a=20

b=30

c=10

val=`expr $a + $b`

echo "a + b : $val"

Val=`expr $a + $b \\* $c`

echo "a + b \* c : $Val"

if(($a == "true" & $b == "true" ))

then

echo Both are true.

else

echo Both are not true.

fi

if(($a == "true" || $b == "true" ))

then

echo Atleast one of them is true.

else

echo None of them is true.

fi

val=`expr $a \\* $b`

echo "a \* b : $val"

val=`expr $b / $a`

echo "b / a : $val"

val=`expr $b % $a`

echo "b % a : $val"

if [ $a == $b ]

then

echo "a is equal to b"

fi

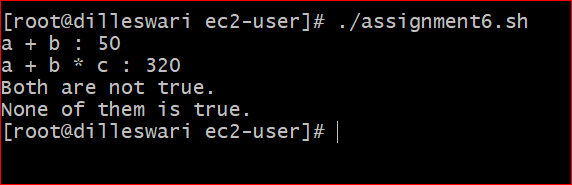
if [ $a != $b ]

then

echo "a is not equal to b"

fi

[ ! -f /etc/foo ] && echo "File does not exist"



Write a script that will use command substitution to dynamically set variable values:

TODAYSDATE - should contain date/time stamp when executed

USERFILES - the results of a find run on the /home directory to list all files owned by 'user' account

Additionally, set two aliases:

TODAY - should be an alias for the 'date' command

UFILES - should be an alias to the full command used to set the variable USERFILES above

Finally, display all variables and alias values when the script is run.

#!/bin/bash

TODAY=$(date)

HOST=$(hostname)

echo "-----------------------------------------------------"

echo "Date: $TODAY Host:$HOST"

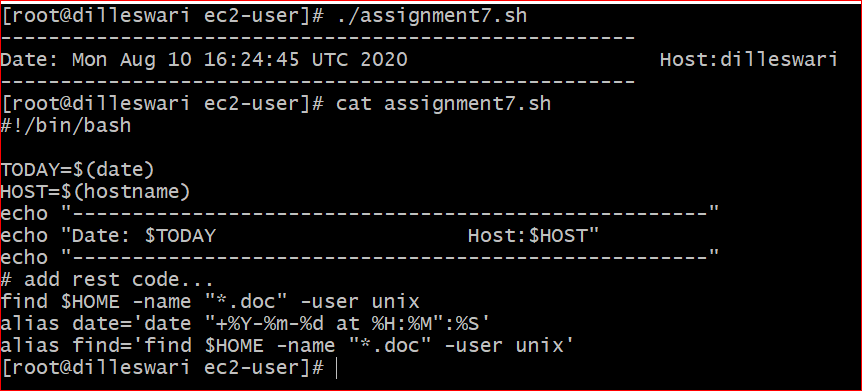
echo "-----------------------------------------------------"

# add rest code...

find $HOME -name "\*.doc" -user laxmi

alias date='date "+%Y-%m-%d at %H:%M":%S'

alias find='find $HOME -name "\*.doc" -user laxmi



8 Create a script that interacts with the user. You will want to prompt the user to enter the following information (which you will capture and place into the following variables):

FIRSTNAME

LASTNAME

USERAGE

Greet the user with their name and current age displayed and then calculate and display their age in 10 years.

#! /bin/bash

# This script display a simple menu, from which user

# can select a choice

# Author: Dilleswari

#Date: aug 10

HORIZONTALLINE="============================================="

CLEAR

echo -e "\n$HORIZONTALLINE"

echo "1) Display system date and time."

echo "2) Display the Calender of the month."

echo "3) Display the Hostname."

echo "4) Display the IP address info for this system."

echo -e "$HORIZONTALLINE\n"

read -p "Type the option you select: "choice

if [ "$choice" -eq "$choice" 2> /dev/null ]; then

if [ $choice -lt l -o $choice -gt 4 ]; then

echo -e "\n==> Enter a number between 1 and 4 <=="

elif [ $choice -eq 1 ]; then

echo -e "\nSystem Date and time: `date`\n"

elif [ $choice -eq 2 ]; then

echo -e "\nCalender of The Month:"

cal

elif [ $choice -eq 3 ]; then

echo -e "\nSystem Hostname: `hostname`\n"

elif [ $choice -eq 4 ]; then

echo -e "\nIP address Info:"

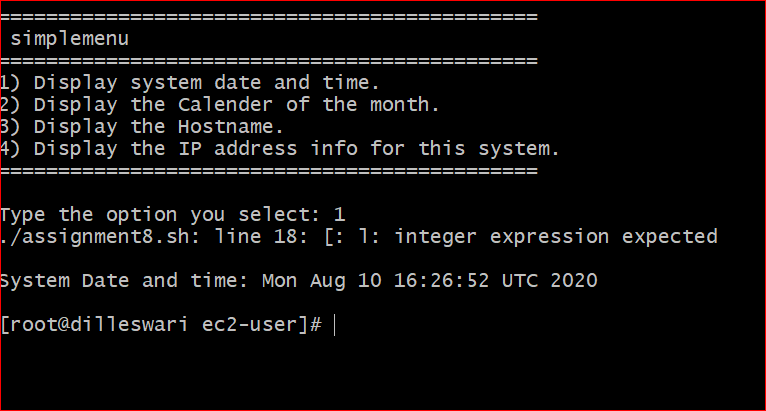
ip a

fi

else

echo -e "\n==> This is not a number <=="

fi



9 Write a script intended to iterate through an array called SERVERLIST containing at least four values (server names). Display all four values to the terminal when run.

#!/bin/bash

SERVERLIST=(host1,host2,host3,host4)

echo "${SERVERLIST[\*]}"

10 Create a script that, when run, will accept two command line values as arguments. These arguments should be a username and password and assigned to two variables in the script named appropriately. Finally, echo those values out to the terminal when run to indicate they were read and assigned as expected.

#!/bin/bash

echo "Enter username"

read n

username=$n

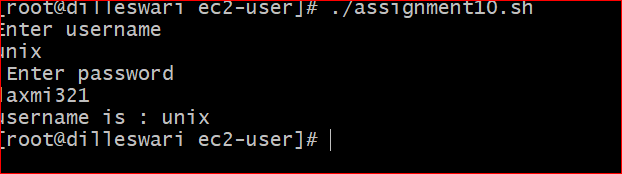
echo " Enter password"

read m

password=$m

echo 'username is :' $username

echo 'password is :' $password



11 Create a script that interacts with the user. Prompt them to guess a secret number between 1 and 5. Read the guess from the terminal and assign it to a variable. Using the 'if' statement from the course, test that value to determine if they guessed the right number (you choose the correct value). If they do, display a success message, otherwise do nothing.

#!/bin/bash

echo 'enter number between 1-5'

read number

secretnumber=5

if [ $number == $secretnumber ]

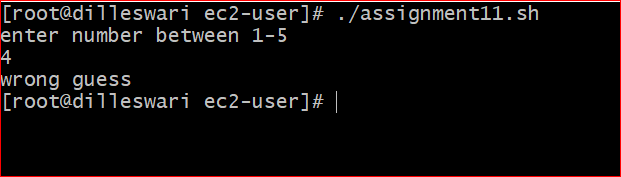
then

echo 'correct guess'

else

echo 'wrong guess'

fi^C



Write a script that will prompt the user to enter a number between 1 and 3. Capture that number in a variable and then test that variable. Be sure to display the variable and it's value as appropriate within an if/then/else statement where the final statement will display if they did not enter a number between 1 and 3 telling them they failed to follow instructions. Redirect errors from each of the tests to /dev/null (to prevent script errors from showing if text is entered for example).

#!/bin/bash

echo "Enter number between 1-3: "

read number

a=1

b=2

c=3

if [ $number == $a ] || [ $number == $b ] || [ $number == $c ]

then

echo "Number is between 1-3 and number is: " $number

else

echo "Number is not between 1-3 and entered number is: " $number

fi

