Assignment-01

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

USER

HOME

HISTCONTROL

TERM

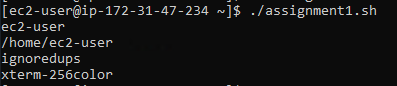
**#! /bin/bash**

**printenv USER**

**printenv HOME**

**printenv HISTCONTROL**

**printenv TERM**



Assignment-02

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`**

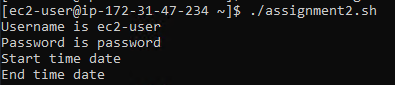
**ENDOFSCRIPT=`date`**

**echo 'username is ' $MYUSERNAME**

**echo 'password is ' $MYPASSWORD**

**echo 'Start time ' $STARTOFSCRIPT**

**echo 'End time ' $ENDOFSCRIPT**



Assignment-03

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 displays a greeting and gives information about currently connected users.**

**# The two example variables are set and displayed.**

**echo "START"**

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

**echo**

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

**echo**

**w # show who is logged on and**

**echo # what they are doing**

**echo "Setting two variables."**

**MYUSERNAME="Admin" # set a local shell variable**

**MYPASSWORD="passwd" # set a local shell variable**

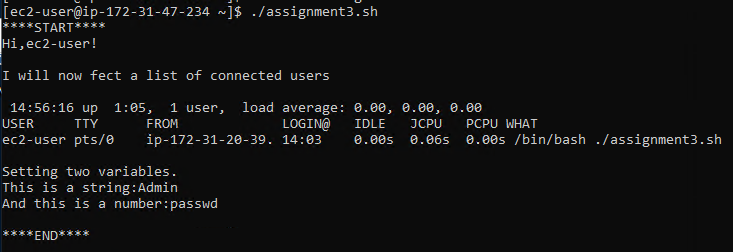
**echo "This is a string: $MYUSERNAME" # display content of variable**

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

**echo**

**echo "END."**

**echo**



Assignment-04

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.

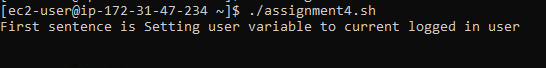
**#! /bin/bash**

**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**



Assignment-05

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"**



Assignment-06

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"**

Assignment-07

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 admin**

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

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

Assignment-08

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: Harshitha**

**#Date: October 06**

**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**

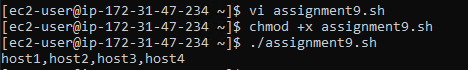
Assignment-09

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[\*]}"**



Assignment-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**

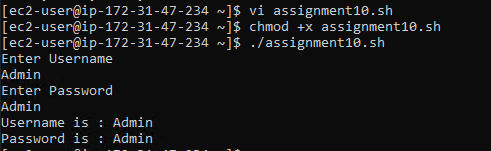
**echo " Enter password"**

**read m**

**password=$m**

**echo 'username is :' $username**

**echo 'password is :' $password**



Assignment-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=3**

**if [ $number == $secretnumber ]**

**then**

**echo 'correct guess'**

**else**

**echo 'wrong guess'**

**fi^C**

Assignment-12

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**

Assignment-13

Write a script that assigns a variable that contains a list of all shell scripts (\*.sh) in the current directory (command redirection). Using the 'for'statement from the course, iterate through that list of files and display the filename of each and cat out the contents to the terminal.

Assignment-14

Develop a simple three item menu to display on the terminal. Your script, upon display of the menu, should prompt the user to choose one of the three available options. Using the 'case' statement from the course, display three unique messages depending on which number they chose, with a catch all message letting them know if they went outside the bounds of instructions.

Assignment-15

Create a script that prompts the user for a number. That number is to be used to display a simple message to the terminal X number of times (where X is the number captured from the user input). The message should include an indication of the number for each count the message is displayed.

Assignment-16

Create a simple text file containing a list of super heros (or some names if preferred), use at least four values, one per line in the file.

Write a bash shell script that then reads that file and displays it line by line on the terminal window.

Assignment-17

Create a simple text file containing a list of names (superheroes) and name it whatever you wish, should contain at least four values, one per line.

Write a script that will use a file descriptor defined for both reading and writing to that file (pick an appropriate number greater than the system reserved handles as talked about in the course). Filtering that file into an appropriate loop, display all values with that file. Finally, once complete, write a message with the time/date stamp to the file and close the descriptor.

Assignment-18

Create a simple text file at the command prompt. This file should contain three values - CPU, Memory and Disk space for an imaginary system, all on one line and delimited with a '|' character.

Write a script to read that file and prompt the user for the delimiter value. Use that delimiter along with the IFS variable and read those delimited values into three appropriately named variables. Finally, display them with labels, one each per line.

Assignment-19

We need to create a menu for our users. Display a menu containing three choices AND a choice to allow them to exit. Display that menu and prompt for a choice. Upon choosing the value, it should simply clear the screen and redisplay the menu (loop until the exit choice is given).

HOWEVER, we need to be sure that the end user cannot use CTL-C, CTL-Z or a KILL command to terminate the application. Add a 'trap' in the script to capture those attempts and provide instructions on how to exit the script using the menu choice instead.

Assignment-20

Create a script that accepts a command line parameter intended to be the name of a directory. With the script, attempt to change to the indicated directory, be sure to redirect errors to /dev/null on the command as we will be providing our own messaging.

Test the results of the command to change directories. If it was successful, indicate success and display the contents of the directory. If it was not successful, indicate we cannot change directories and exit to the terminal with a custom exit code (less than 200).

Assignment-21

Create a simple script containing a single function. This function should display a message to clearly indicate it was generated when the function was run. Then, display another message outside the function clearly indicating it was generated outside of it.

Assignment-22

Create a script to demonstrate the visibility of variables and when they are available within a script and its functions. Define a global variable, a function that defines a local variable and then display both BEFORE calling the function, call the function, then display both AFTER calling the function.

Assignment-23

Write a script that takes a single command line parameter intended to be the user's first name. Prompt the user for their age and read that into a variable. Using the appropriate method to make that command line parameter visible to a function, pass the age captured to the function and display a message to the user addressing them by name and confirming their age, add a calculation of their age in number of days.

Assignment-24

We are going to use nested functions to simulate the kind of abstraction you find in many object oriented languages. Create the following structures in your script:

a function that defines two local variables to hole the number of arms and legs that a human being has.

nested functions, one for each a male and female, that contain the appropriate number of beards that each gender has.

capture the gender as a command line parameter.

test the command line parameter and call the appropriate functions in order to display the characteristics of the indicated gender.

Assignment-25

We want to display a simple Information Box for our end users prior to executing a command. Accept one command line parameter when executing the script. This box should use the dialog control as shown in the course and display for a total of 5 seconds. The title and message in the box should be passed into the function but can be whatever you like that will warn the user if the parameter passed in was 'shutdown', otherwise an innocuous message can be displayed.

Assignment-26

We want to display a simple Message Box for our end users prior to executing a command. Accept one command line parameter when executing the script. This box should use the dialog control as shown in the course and display until the OK button is clicked or selected. The title and message in the box should be passed into the function but can be whatever you like that will warn the user if the parameter passed in was 'shutdown', otherwise an innocuous message can be displayed.

Assignment-27

In this script, we will be using an Input Box constructed from the dialog control, to prompt the user for a filename to display to the terminal. Construct the input box within a function and capture the value input using the appropriate output method to a file. Read that file back in and attempt to display (cat) the file to the terminal or indicate that it does not exist.

Assignment-28

Using the dialog control from the course, develop a function inside a script that will display a menu containing at least four choices. Capture the indicated value using the appropriate output to a file. Reading that file, test the value and display an appropriate message, different for each one.