# Homework 4: Introduction to Bash Scripting in Linux

This homework focuses on providing you with an introduction to the Bash scripting language common in Unix/Linux operating systems.

Requirements: Ubuntu Linux or Kali Linux. SElinux is acceptable, but some commands may be different.

If you cannot set up a Ubuntu or Kali VM, please work with a team partner who has access to a Ubuntu or Kali VM. Ubuntu, Kali Linux, and VirtualBox are all free to download and install:

Kali Linux:

<https://www.kali.org/get-kali/>

Ubuntu Linux:

<https://ubuntu.com/#download>

Oracle VirtualBox:

<https://www.virtualbox.org/>

# Intro to Bash

Launch a terminal in Linux.

1. You should be in your home directory at the launch of your terminal. Type **pwd**. Hit enter. The output should appear something like /home/username/.
2. Now we want to create our first bash file. To do this, we’re going to use the built in vi editor. Type in **vi hello.sh** Hit enter.
3. A vi prompt should appear. Start typing to enter edit mode. On the first line, type

**echo -e “Hello World\n”** . Hit enter. What do you think echo does?

1. Now we need to save our changes, this can be a bit tricky if you’ve never used vi before. We need to exit out of edit mode and get back into the command line mode. Hit **escape** once. Type in **:wq!** (Your prompt should be at the bottom of the command line). Hit enter. You should be back at your regular terminal.
2. What do you think :wq! does? Test to see if the command worked by checking the hello.sh document in the vi editor.
3. Type ls to see if the hello.sh file appears. If so, great! You just created a bash file! Take a screenshot of your home directory following the ls command.
4. Type **ls -l** to view the current permission for the hello.sh file.
5. What would we type in to grant full permissions to everyone for this file? (look in the Linux and Unix OS chapter if you aren’t sure). Type in the correct command and hit enter. Take a screenshot of the hello.sh file with full read, write, and execute permissions for users, groups, and others.
6. Now that you have permission to execute the file, lets run it. Type in **./hello.sh** Hit enter. What was the output? Based on this output what do you think the Echo Command does? Why do you also think we added in the “\n” at the end of the string”?
7. Now, re-edit the hello.sh file by entering the vi editor
8. Type in a new line to echo out the following statement: **What University do you attend?**
9. Now we want to prompt the user for some sort of input after this statement. To prompt a user we need to use the command **read**, then we need to specify a variable to store the user’s input. Go to the line below your last echo statement and type in **read uni** Hit enter.
10. Now we want to print out that input back to the user as output. To do that, go to another line and type **echo -e “$uni”**. Hit enter.
11. Exit out of the vi editor, and make sure you save your changes (go back to step 4 & 5 if you don’t remember how)
12. Re run the shell script hello.sh (same procedure as in step 9), but now notice that the script is waiting for input. Type in **WVU or West Virginia University**, and hit enter. What happened?

# Basic Conditional Statements

1. Create a new bash file called university.sh
2. In this bash file, echo out a statement **“What University do you attend?”**, add in a user prompt using the read command, and store that input variable as **uni**
3. We now need to create a conditional “if” statement. (spaces matter in bash) Type in:

**if [ “$uni” == “WVU” ] || [ “$uni” == “West Virginia University” ];**

**then**

**echo -e “Welcome to West Virginia University \n”**

**else**

**echo -e “Access Denied\n”**

**fi**

1. Save your bash code, and then try to run the university.sh shell script. (If it doesn’t work initially, think about why! \*HINT – remember your file access permissions)
2. What happens when you enter in WVU? What happens when you enter in West Virginia University? What happens when you enter in Pitt or OSU?
3. Why do you think both of the WVU and West Virginia University inputs works? Where is the operator that makes both of these inputs work?
4. See if you can modify this bash script to have multiple if statements that accept input from another university in the prompt, in addition to WVU. A good example would be Marshall university, with an accompanying message (i.e. “Welcome to Marshall University”). Look up “else if” or “elif” functions in bash to help you modify this code. Post your code and a screenshot of the outputs below.

# Ping Tool in Bash

1. Create a new bash file called ping.sh
2. Echo out an output “What would you like to ping”
3. Create an input and store that input as the variable **var**
4. Then type the ping command **ping -c 4 $var**
5. Save your bash file and assign yourself execute permissions.
6. Run the bash file and specify an IP address or domain name and ping that address.
7. Take a screenshot of the output
8. Modify your shell script to prompt the user to also specify the number of times you want to ping. Paste a screenshot of your modified shell script and your code below.

# SystemInfo Script in Bash

This lesson is adapted from: <https://www.2daygeek.com/bash-shell-script-view-linux-system-information/>

1. Take a look at the shell script on this site. Create a new shell file and try to copy in this code and make it work.

#!/bin/bash

echo -e "-------------------------------System Information----------------------------"

echo -e "Hostname:\t\t"`hostname`

echo -e "uptime:\t\t\t"`uptime | awk '{print $3,$4}' | sed 's/,//'`

echo -e "Manufacturer:\t\t"`cat /sys/class/dmi/id/chassis\_vendor`

echo -e "Product Name:\t\t"`cat /sys/class/dmi/id/product\_name`

echo -e "Version:\t\t"`cat /sys/class/dmi/id/product\_version`

echo -e "Serial Number:\t\t"`cat /sys/class/dmi/id/product\_serial`

echo -e "Machine Type:\t\t"`vserver=$(lscpu | grep Hypervisor | wc -l); if [ $vserver -gt 0 ]; then echo "VM"; else echo "Physical"; fi`

echo -e "Operating System:\t"`hostnamectl | grep "Operating System" | cut -d ' ' -f5-`

echo -e "Kernel:\t\t\t"`uname -r`

echo -e "Architecture:\t\t"`arch`

echo -e "Processor Name:\t\t"`awk -F':' '/^model name/ {print $2}' /proc/cpuinfo | uniq | sed -e 's/^[ \t]\*//'`

echo -e "Active User:\t\t"`w | cut -d ' ' -f1 | grep -v USER | xargs -n1`

echo -e "System Main IP:\t\t"`hostname -I`

echo ""

echo -e "-------------------------------CPU/Memory Usage------------------------------"

echo -e "Memory Usage:\t"`free | awk '/Mem/{printf("%.2f%"), $3/$2\*100}'`

echo -e "Swap Usage:\t"`free | awk '/Swap/{printf("%.2f%"), $3/$2\*100}'`

echo -e "CPU Usage:\t"`cat /proc/stat | awk '/cpu/{printf("%.2f%\n"), ($2+$4)\*100/($2+$4+$5)}' | awk '{print $0}' | head -1`

echo ""

echo -e "-------------------------------Disk Usage >80%-------------------------------"

df -Ph | sed s/%//g | awk '{ if($5 > 80) print $0;}'

echo ""

1. Run the shell script, and paste a screenshot of the output.
2. What commands worked and what commands failed?
3. For any failed commands, can you try to get them to work? Are there permissions issues with the files?

# Network Diagnostic Tool

1. Using your previous bash scripts as a template, write a brief bash script that displays some basic networking information about your system using common linux commands (i.e. ifconfig) (you can be creative as you like with this!).
2. Run your shell script and post a screenshot of the output as well as your code below.