Shell Scripting
Set 1

Rwithik Manoj

March 22, 2019

1. Write a shell script to show various system configuration like

   (a) Currently logged user and his login name

   (b) Your current shell

   (c) Your home directory

   (d) Your operating system type

   (e) Your current path setting

   (f) Your current working directory

   (g) Number of users currently logged in

   Algorithm:

   (a) Start

   (b) Print required configurations using system variables

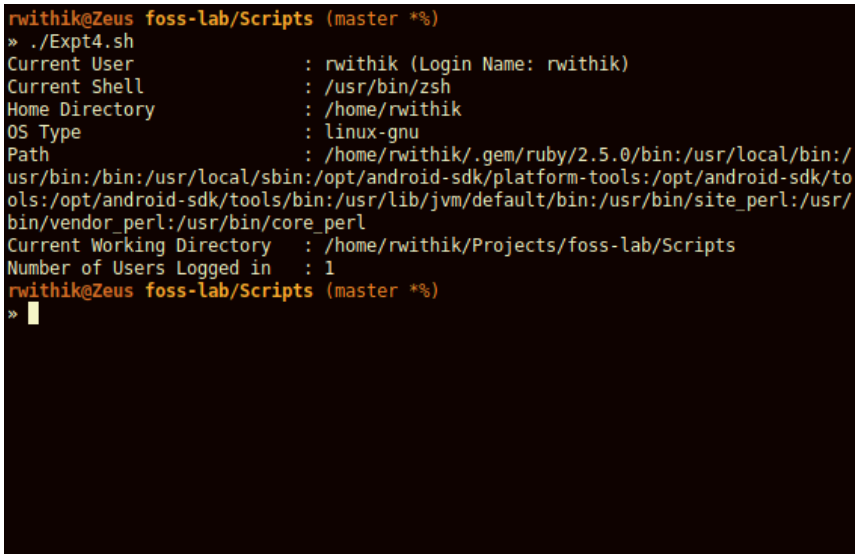
   (c) Stop

   Script:

```
#!/bin/bash

echo -e "Current User                : $USER (Login Name: $LOGNAME)"
echo -e "Current Shell               : $SHELL"
echo -e "Home Directory              : $HOME"
echo -e "OS Type                     : $OSTYPE"
echo -e "Path                        : $PATH"
echo -e "Current Working Directory   : $PWD"
echo -e "Number of Users Logged in   : `who | wc -l`"
```

   Output:



2. Write a shell script to show various system configurations like

   - Your OS and version, release number, kernel version
   - All available shells
   - Computer CPU information like processor type, speed etc
   - Memory information
   - Hard disk information like size of hard-disk, cache memory, model etc
   - File system (Mounted)

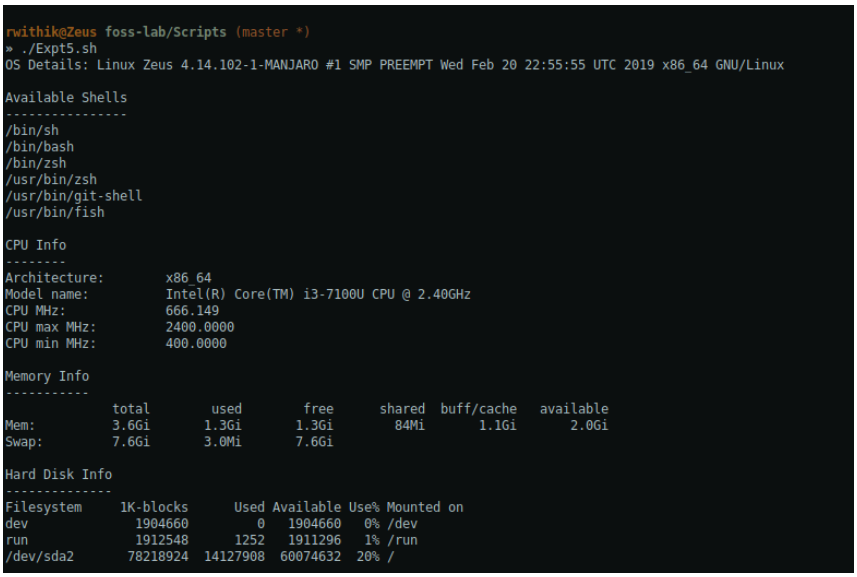
   (a) Start
   (b) Print the OS details using uname command
   (c) Print the availabel shells from /etc/shells
   (d) Print CPU info using lscpu command
   (e) Print the memory info using free command
   (f) Print the hard disk info using df command
   (g) Print the filesystems using lsblk command
   (h) Stop

   Script:

```
#!/bin/bash

echo -e "OS Details: `uname -a`\n"
echo -e "Available Shells"
echo -e "----------------"
cat /etc/shells | grep -e "^/"
echo -e "\nCPU Info"
echo -e "--------"
lscpu | grep "Architecture\|Model name\|MHz"
echo -e "\nMemory Info"
echo -e "-----------"
free -mh
echo -e "\nHard Disk Info"
echo -e "--------------"
df
echo -e "\nMounted Filesystems"
echo -e "-------------------"
lsblk
```

   Output:


rwithik@Zeus foss-lab/Scripts (master *)
» ./Expt5.sh
OS Details: Linux Zeus 4.14.102-1-MANJARO #1 SMP PREEMPT Wed Feb 20 22:55:55 UTC 2019 x86_64 GNU/Linux
Available Shells
/bin/sh
/bin/bash
/bin/zsh
/usr/bin/zsh
/usr/bin/git-shell
/usr/bin/fish
CPU Info
Architecture: x86_64
Model name: Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz
CPU MHz: 666.149
CPU max MHz: 2400.0000
CPU min MHz: 400.0000
Memory Info
total used free shared buff/cache available
Mem: 3.6Gi 1.3Gi 1.3Gi 84Mi 1.1Gi 2.0Gi
Swap: 7.6Gi 3.0Mi 7.6Gi
Hard Disk Info
Filesystem 1K-blocks Used Available Use% Mounted on
dev 1904660 0 1904660 0% /dev
run 1912548 1252 1911296 1% /run
/dev/sda2 78218924 14127908 60074632 20% /

3. Write a shell script to implement a menu driven calculator with following functions

   (a) Addition

   (b) Subtraction

   (c) Multiplication

   (d) Division

   (e) Modulus

   Algorithm:

   (a) Start

   (b) Read the operations

   (c) Check if the operation is valid

   (d) If valid, perform the operation, else print error message

   (e) Stop

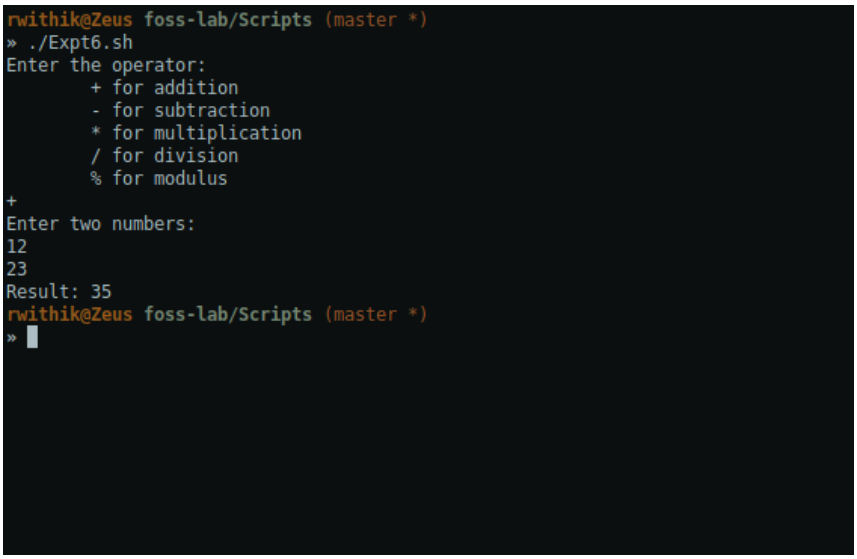
   Script:

```
#!/bin/bash

echo -e "Enter the operator:\n\
                + for addition\n\
                - for subtraction\n\
                * for multiplication\n\
                / for division\n\
                % for modulus"

read CHOICE
if [[ $CHOICE == '+' ]] || [[ $CHOICE == '-' ]]\
                || [[ $CHOICE == '*' ]] || [[ $CHOICE == '/' ]]\
                || [[ $CHOICE == '%' ]]
then
        echo -e "Enter two numbers:"
        read A
        read B
        echo -e "Result: $[${A}${CHOICE}${B}]"
else
        echo -e "Invalid Choice"
fi
```

   Output:


rwithik@Zeus foss-lab/Scripts (master *)
» ./Expt6.sh
Enter the operator:
+ for addition
- for subtraction
* for multiplication
/ for division
% for modulus
+
Enter two numbers:
12
23
Result: 35
rwithik@Zeus foss-lab/Scripts (master *)
»

4. Write a script called addnames that is to be called as follows ./addnames ulist username. Here ulist is the name of the file that contains list of user names and username is a particular student's username. The script should

   (a) Check that the correct number of arguments was received and print a message, in case the number of arguments is incorrect

   (b) Check whether the ulist file exists and print an error message if it does not

   (c) Check whether the username already exists in the file. If the username exists, print a message stating that the name already exists. Otherwise, add the username to the end of the list.

   Algorithm:

   (a) Start

   (b) Check if the number of arguments is equal to two

   (c) Check if the file exists

   (d) Iterate through the file and check if the given name exists in the file

   (e) If it doesn't exist in the file, add the name to the file

   (f) Stop:

   Script:

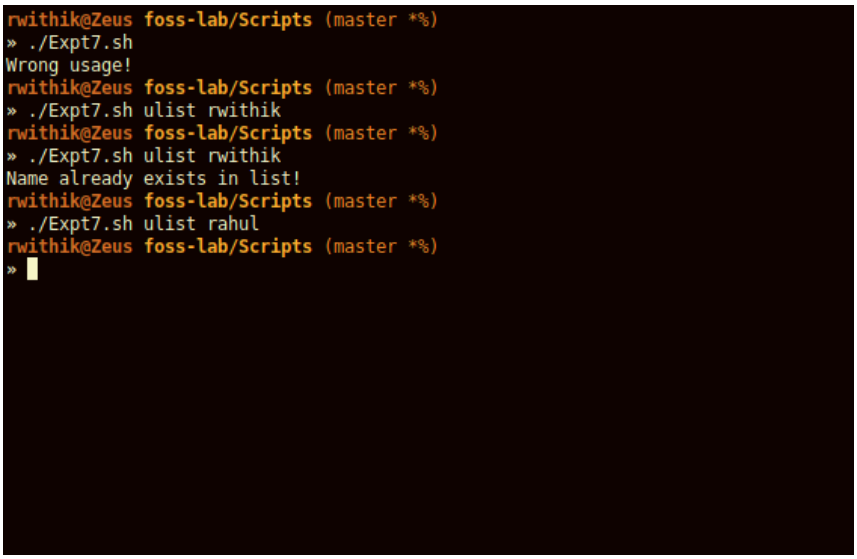
```
#!/bin/bash

if [[ $# -ne 2 ]]; then
        echo -e "Wrong usage!"
        exit
fi

if [[ -f $1 ]]; then
        for name in `cat $1`; do
                if [[ $name == $2 ]]; then
                        echo -e "Name already exists in list!"
                        exit
                fi
        done
        echo $2 >> $1
        exit
fi

echo -e "File $1 not found!"
```

   Output:


rwithik@Zeus foss-lab/Scripts (master *%)
» ./Expt7.sh
Wrong usage!
rwithik@Zeus foss-lab/Scripts (master *%)
» ./Expt7.sh ulist rwithik
rwithik@Zeus foss-lab/Scripts (master *%)
» ./Expt7.sh ulist rwithik
Name already exists in list!
rwithik@Zeus foss-lab/Scripts (master *%)
» ./Expt7.sh ulist rahul
rwithik@Zeus foss-lab/Scripts (master *%)
»

5. Write a Shell script which starts on system boot up and kills every process which uses more than a specified amount of memory or CPU.
Algorithm:

   (a) Start

   (b) Read inputs from a ps command.

   (c) Iterate through the inputs

   (d) Check if the process belong to the root user. If it does, skip it.

   (e) Check if the process uses more than the specified amount of CPU or memory. If it does, kill it using kill command

   (f) Stop

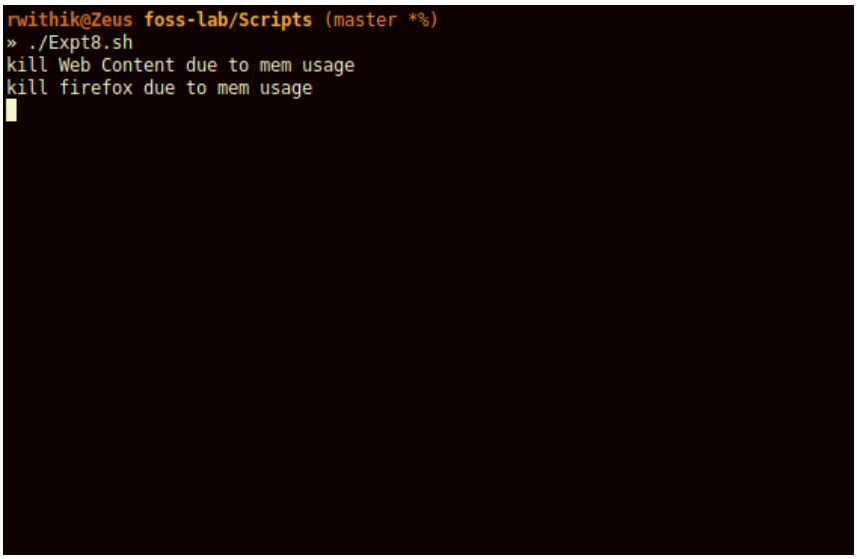
Script:

```
#!/bin/bash

CHECK_MEM_VAL=10.0
CHECK_CPU_VAL=10.0


while true
do
        ps -e -o pmem=,pcpu=,pid=,user=,comm= --sort=-pmem |
          while read SIZE CPU PID USER COMM
          do
            KILL_MEM=0
            KILL_CPU=0
            if [ "$USER" != "root" ]
            then
              if [ `echo "$SIZE>$CHECK_MEM_VAL" | bc` = "1" ]
              then
                        echo "kill $COMM due to mem usage"
                kill $pid
              elif [ `echo "$CPU>$CHECK_CPU_VAL" | bc` = "1" ]
              then
                echo "kill $COMM due to cpu usage"
                kill $pid
              fi
            fi
        done
        sleep 10
done
```

Output:


rwithik@Zeus foss-lab/Scripts (master *%)
» ./Expt8.sh
kill Web Content due to mem usage
kill firefox due to mem usage