***Exercise\_1 - Write a shell script that prints “Shell Scripting is Fun!” on the screen***

#!/bin/bash

echo “Shell Scripting is Fun!”

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe1.sh  
Shell Scripting is Fun!

***Exercise\_2 - Modify the shell script from exercise 1 to include a variable. The variable will hold the contents of the message “Shell Scripting is Fun!”***

#!/bin/bashNAME=”Shell Scripting is Fun!”  
echo $NAME

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe2.sh  
Shell Scripting is Fun!

***Exercise\_3 - Store the output of the command “hostname” in a variable. Display “This script is running on \_.” where “\_” is the output of the “hostname” command.***

#!/bin/bashHOSTNAME=$(hostname)  
echo “This script is running on $HOSTNAME”

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe3.sh  
This script is running on vimukthi

***Exercise\_4 - Write a shell script to check to see if the file “file\_path” exists. If it does exist, display “file\_path passwords are enabled.” Next, check to see if you can write to the file. If you can, display “You have permissions to edit “file\_path.””If you cannot, display “You do NOT have permissions to edit “file\_path””***

#!/bin/bashFILE=”/home/svimukthi/Assignment/sanka”

if [ -e “$FILE” ]  
 then  
 echo “$FILE passwords are enabled”  
fiif [ -x “$FILE” ]  
 then  
 echo “You have permition to execute $FILE” else  
 echo “You do Not have permissions to execute $FILE”  
fi

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe4.sh  
/home/svimukthi/Assignment/sanka passwords are enabled  
You have permition to execute /home/svimukthi/Assignment/sanka

***Exercise\_5 - Write a shell script that displays “man”,”bear”,”pig”,”dog”,”cat”,and “sheep” on the screen with each appearing on a separate line. Try to do this in as few lines as possible.***

#!/bin/bash

ANIMALS=”man bear pig dog cat sheep”

for ANIMAL in $ANIMALS  
 do  
 echo $ANIMAL  
 done

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe5.sh  
man  
bear  
pig  
dog  
cat  
sheep

***Exercise\_6 - write a shell script that prompts the user for a name of a file or directory and reports if it is a regular file, a directory, or another type of file. Also perform an ls command against the file or directory with the long listing option.***

#!/bin/bash

echo “Enter the file path”  
read FILEif [ -f “$FILE” ]  
 then  
 echo “$FILE is a reguler file”elif [ -d “$FILE” ]  
 then  
 echo “$FILE is a directory”else  
 echo “$FILE is another type of file”  
fils -l $FILE

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe6.sh  
Enter the file path  
/home/svimukthi/sanka  
/home/svimukthi/sanka is a directory  
total 4  
drwxr-xr-x. 2 svimukthi svimukthi 30 Nov 13 20:18 hs  
-rw-rw-r — . 1 svimukthi svimukthi 12 Nov 12 12:09 sanka.txt  
d — x — — — . 2 svimukthi svimukthi 20 Dec 13 18:53 test

***Exercise\_7 - Modify the previous script to that it accepts the file or directory name as an argument instead of prompting the user to enter it.***

#!/bin/bash

FILE=$1if [ -f “$FILE” ]  
 then  
 echo “$FILE is a reguler file”elif [ -d “$FILE” ]  
 then  
 echo “$FILE is a directory”else  
 echo “$FILE is another type of file”  
fils -l $FILE

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe7.sh /home/svimukthi/sanka  
/home/svimukthi/sanka is a directory  
total 4  
drwxr-xr-x. 2 svimukthi svimukthi 30 Nov 13 20:18 hs  
-rw-rw-r — . 1 svimukthi svimukthi 12 Nov 12 12:09 sanka.txt  
d — x — — — . 2 svimukthi svimukthi 20 Dec 13 18:53 test

***Exercise\_8 - Modify the previous script to accept an unlimited number of files and directories as arguments.***

#!/bin/bash

FILES=$@for FILE in $FILES  
 do  
 if [ -f “$FILE” ]  
 then  
 echo “$FILE is a reguler file” elif [ -d “$FILE” ]  
 then  
 echo “$FILE is a directory” else  
 echo “$FILE is another type of file”  
 fi ls -l $FILE done

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe8.sh /home/svimukthi/sanka /hms/apps  
/home/svimukthi/sanka is a directory  
total 4  
drwxr-xr-x. 2 svimukthi svimukthi 30 Nov 13 20:18 hs  
-rw-rw-r — . 1 svimukthi svimukthi 12 Nov 12 12:09 sanka.txt  
d — x — — — . 2 svimukthi svimukthi 20 Dec 13 18:53 test  
/hms/apps is a directory  
total 60  
drwx — — — . 7 svimukthi svimukthi 4096 Dec 7 15:08 ajuba  
drwx — — — . 7 svimukthi svimukthi 4096 Dec 7 15:08 ajuba-preference-data-loader  
drwx — — — . 7 svimukthi svimukthi 4096 Nov 30 10:25 ajuba-survey  
-rw-rw-r — . 1 svimukthi svimukthi 22772 Dec 10 17:01 sendKindle-2.1–6.el7.psychotic.noarch.rpm  
-rw-rw-r — . 1 svimukthi svimukthi 19875 Dec 10 17:04 sendKindle-2.1–6.el7.psychotic.src.rpm  
drwxr-xr-x. 2 svimukthi svimukthi 4096 Dec 17 08:24 Versions\_Script

***Exercise\_9 - Write a shell script that displays, “This script will exit with 0 exit status.” Be sure that the script does indeed exit with a 0 exit status.***

#!/bin/bash

echo “This script will exit with 0 exit status.”  
exit 0

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe9.sh  
This script will exit with 0 exit status.

***Exercise\_10 - Write a shell script that accepts a file or directory name as an argument. Have the script report if it is reguler file, a directory, or another type of file. If it is a directory, exit with a 1 exit status. If it is some other type of file, exit with a 2 exit status.***

#!/bin/bash

FILE=$1if [ -f $FILE ]  
 then  
 echo “It is reguler File”  
 exit 0elif [ -d $FILE ]  
 then  
 echo “It is directory”  
 exit 1 else  
 echo “Another type”  
 exit 2  
fi

**output**

[svimukthi@sanka Shell\_Scripting]$ ./exe10.sh /home/svimukthi/sanka  
It is directory

***Exercise\_11 - Write a script that executes the command “cat/etc/shadow”. If the command return a 0 exit status, report “command succeeded” and exit with a 0 exit status. If the command returns a non-zero exit status, report “Command failed” and exit with a 1 exit status.***

#!/bin/bash

cat /etc/shadowif [ “$?” -eq “0” ]  
 then  
 echo “Command succeeded”  
 exit 0 else  
 echo “Command failed”  
 exit 1  
fi

**Output**

[svimukthi@sanka Shell\_Scripting]$ ./exe11.sh  
cat: /etc/shadow: Permission denied  
Command failed

***Exercise\_12 - Write a shell script that consists of a function that displays the number of files in the present working directory. Name this function “file\_count” and call it in your script. If you use variable in your function, remember to make it a local variable.***

#!/bin/bash

function file\_count()  
 {  
 local NUMBER\_OF\_FILE=$(ls -l | wc -l)  
 echo "$NUMBER\_OF\_FILE"  
 }

file\_count

**Output**

[vimukthi@vimukthi Test]$ ./coun.sh   
4

***Exercise\_13 - Modify the script from the previous exercise. Make the “file\_count” function accept a directory as an argument. Next, have the function display the name of the directory followed by a colon. Finally display the number of files to the screen on the next line. Call the function three times. First on the “/etc” directory, next on the “/var” directory and finally on the “/usr/bin” directory.***

#!/bin/bash

function file\_count()  
 {  
 local Directory=$1  
 COUNT\_FILE=$(ls $Directory|wc -l)  
 echo "$Directory"  
 echo "$COUNT\_FILE"  
 }file\_count /etc  
file\_count /var  
file\_count /usr/bin

**Output**

[vimukthi@vimukthi Test]$ ./exe13.sh  
/etc  
295  
/var  
22  
/usr/bin  
2020

***Exercise\_14 - Write the shell script that renames all files in the current directory that end in “.jpg” to begin with today’s date in the following format: YYYY-MM-DD. For example, if a picture of my cat was in the current directory and today was October 31,2016 it would change name from “mycat.jpg” to “2016–10–31-mycat.jpg”.***

#!/bin/bash

DAY=$(date +%F)  
cd /home/vimukthi/Picturesfor FILE in \*.png  
 do  
 mv $FILE ${DAY}-${FILE}  
 done

After executing that script we can see change the names.

***Exercise\_15 - Write the script that renames files based on the file extension. Next,It should ask the user what prefix to prepend to the file name(s). By default, the prefix should be the current date in YYYY-MM-DD format. If the user simply press enter,the current date will be used. Otherwise,whatever the user entered will be used as the prefix. Next,it should display the original file name and new name of the file. Finally,it should rename the file.***

#!/bin/bash

cd /home/vimukthi/Pictures  
DAY=$(date +%F)secho "Pleace enter the file extension:"  
 read EXTENSIONecho "Pleace enter the prifix:(press enter for $DAY)"  
 readfor NAME in \*.$EXTENSION  
 do  
 echo "Renaming $NAME to ${DAY}-${NAME}"  
 mv $NAME ${DAY}-${NAME}  
 done

After executing that script we can see change the names.

***Exercise\_16 - Created the start-up script for an application start and stop.***

#!/bin/bash

INPUT=$1  
cd /hms/installs/mongod/mongodb-linux-x86\_64-2.6.0/bin  
case $INPUT instart)  
 ./mongod -f ../../mongod.conf &  
 echo "Mongodb server Start"  
 ;;stop)  
 PID\_ID=$(ps -ef | grep mongo | cut -d" " -f3 | sed '1!d')  
 kill $PID\_IDif [ $? -eq '0']  
 echo "Mongodb server Stop"  
 ;;\*)  
 echo "Error input"  
 ;;esac

***Exercise\_17 - Write the shell script that displays one random number on the screen and also generates a system log message with that random number.Use the “user” facility and “info” facility for your messages.***

#!/bin/bashMESSAGE="Random number is:$RANDOM"  
echo "$MESSAGE"logger -p user.info "$MESSAGE"

**Run and check log**

[root@vimukthi Test]# ./test.sh   
Random number is:13461  
[root@vimukthi Test]# cat /var/log/messages | tail -n1  
Jan 10 11:03:18 vimukthi vimukthi: Random number is:13461

***Exercise\_18 - Modify the previous script to that it uses a logging function. Additionally, tag each syslog message with “randomly” and include process ID. Generate a 3 random numbers.***

#!/bin/bash

function logging()  
 {  
 MESSAGE=$@  
 SET\_MESSAGE="Random Number is:$MESSAGE"  
 echo "$SET\_MESSAGE"  
 logger -i -p user.info "$SET\_MESSAGE"  
 }logging $RANDOM  
logging $RANDOM  
logging $RANDOM

**Run and check log**

[root@vimukthi Test]# ./exe18.sh  
Random Number is:22009  
Random Number is:25546  
Random Number is:29800  
[root@vimukthi Test]# cat /var/log/messages | tail -n5  
Jan 10 11:20:01 vimukthi systemd: Removed slice User Slice of root.  
Jan 10 11:20:01 vimukthi systemd: Stopping User Slice of root.  
Jan 10 11:20:03 vimukthi vimukthi[6210]: Random Number is:22009  
Jan 10 11:20:03 vimukthi vimukthi[6211]: Random Number is:25546  
Jan 10 11:20:03 vimukthi vimukthi[6212]: Random Number is:29800

***Exercise\_19 - Write a shell script that exits on error and displays command as they will execute, including all expansions and substitutions. Use 3 ls command in your script. Make the first one succeed, the second one fail, and third one succeed. If you are using the proper options, the third ls command not be executed.***

#!/bin/bash

-exls /hmsls /hms/msls /hms/apps

**Output**

[vimukthi@vimukthi Test]$ ./test.sh   
+ ls /hms  
apps backups installs logs support Test  
+ ls /hms/ms  
ls: cannot access /hms/ms: No such file or directory

***Exercise\_20 - Modify the previous exercise so that script continuous, even if an error occurs. This time, all three ls command will execute.***

#!/bin/bash

-xls /hmsls /hms/msls /hms/apps

**Output**

[vimukthi@vimukthi Test]$ ./test.sh   
+ ls /hms  
apps backups installs logs support Test  
+ ls /hms/ms  
ls: cannot access /hms/ms: No such file or directory  
+ ls /hms/apps  
ajuba ajuba-preference-data-loader ajuba-survey

#!/bin/bash

# Print default output

echo `date`

# Print current date without the time

echo `date +"%m-%d-%y"`

# Use 4 digits for year

echo `date +"%m-%d-%Y"`

# Display time only

echo `date +"%T"`

# Display 12 hour time

echo `date +"%r"`

# Time without seconds

echo `date +"%H:%M"`

# Print full date

echo `date +"%A %d %b %Y %H:%M:%S"`

# Nanoseconds

echo Nanoseconds: `date +"%s-%N"`

# Different timezone by name

echo Timezone: `TZ=":US/Eastern" date +"%T"`

echo Timezone: `TZ=":Europe/UK" date +"%T"`

# Print epoch time - convenient for filenames

echo `date +"%s"`

# Print week number

echo Week number: `date +"%V"`

# Create unique filename

f=`date +"%s"`

touch $f

ls -l $f

rm $f

# Add epoch time to existing file

f="/tmp/test"

touch $f

mv $f $f.`date +"%s"`

ls -l "$f".\*

rm "$f".\*

|  |  |
| --- | --- |
| 2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #!/bin/bash  **function** f1 {  echo Hello from $FUNCNAME!  VAR="123"  }  f2() {  p1=$1  p2=$2  sum=**$((**${p1} + ${p2}**))**  echo "${sum}"  }  f1  echo ${VAR}  mySum="**$(**f2 1 2**)**"  echo mySum = $mySum  mySum="**$(**f2 10 -2**)**"  echo mySum = $mySum |

Watch free disk space

#!/bin/bash

# default value to use if none specified

PERCENT=30

# test for command line arguement is present

**if** [[ $# -le 0 ]]

**then**

printf "Using default value for threshold!\n"

# test if argument is an integer

# if it is, use that as percent, if not use default

**else**

**if** [[ $1 =~ ^-?[0-9]+([0-9]+)?$ ]]

**then**

PERCENT=$1

**fi**

**fi**

let "PERCENT += 0"

printf "Threshold = %d\n" $PERCENT

df -Ph | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5,$1 }' | **while** read data;

**do**

used=**$(**echo $data | awk '{print $1}' | sed s/%//g**)**

p=**$(**echo $data | awk '{print $2}'**)**

**if** [ $used -ge $PERCENT ]

**then**

echo "WARNING: The partition \"$p\" has used $used% of total available space - Date: **$(**date**)**"

**fi**

**done**

Rotating log files

#!/bin/bash

f="/home/mtsouk/connections.data"

**if** [ ! -f $f ]

**then**

echo $f does not exist!

exit

**fi**

touch ${f}

MAXSIZE=**$((**4096\*1024**))**

size=`du -b ${f} | tr -s '\t' ' ' | cut -d' ' -f1`

**if** [ ${size} -gt ${MAXSIZE} ]

**then**

echo Rotating!

timestamp=`date +%s`

mv ${f} ${f}.$timestamp

touch ${f}

**fi**

Monitoring the Number of TCP Connections[Permalink](https://www.linode.com/docs/development/bash/solving-real-world-problems-with-bash-scripts-a-tutorial/#monitoring-the-number-of-tcp-connections)

|  |  |
| --- | --- |
| 2  3  4  5  6 | #!/bin/bash  C=**$(**/bin/netstat -nt | tail -n +3 | grep ESTABLISHED | wc -l**)**  D=**$(**date +"%m %d"**)**  T=**$(**date +"%H %M"**)**  printf "%s %s %s\n" "$C" "$D" "$T" |

### Sorting in bash[Permalink](https://www.linode.com/docs/development/bash/solving-real-world-problems-with-bash-scripts-a-tutorial/#sorting-in-bash)

#!/bin/bash

# test that at least one argument was passed

**if** [[ $# -le 0 ]]

**then**

printf "Not enough arguments!\n"

exit

**fi**

count=1

**for** arg in "$@"

**do**

**if** [[ $arg =~ ^-?[0-9]+([0-9]+)?$ ]]

**then**

n[$count]=${arg}

let "count += 1"

**else**

echo "$arg is not a valid integer!"

**fi**

**done**

sort -n <(printf "%s\n" "${n[@]}")

### A Game Written in bash[Permalink](https://www.linode.com/docs/development/bash/solving-real-world-problems-with-bash-scripts-a-tutorial/#a-game-written-in-bash)

#!/bin/bash

NUMGUESS=0

echo "$0 - Guess a number between 1 and 20"

(( secret = RANDOM % 20 + 1 ))

**while** [[ guess -ne secret ]]

**do**

(( NUMGUESS = NUMGUESS + 1 ))

read -p "Enter guess: " guess

**if** (( guess < $secret )); **then**

echo "Try higher..."

**elif** (( $guess > $secret )); **then**

echo "Try lower..."

**fi**

**done**

printf "Yes! You guessed it in $NUMGUESS guesses.\n"

Calculate letter frequency

#!/bin/bash

**if** [ -z "$1" ]; **then**

echo "Usage: $0 filename."

exit 1

**fi**

filename=$1

**while** read -n 1 c

**do**

echo "$c"

**done** < "$filename" | grep '[[:alpha:]]' | sort | uniq -c | sort -nr

Timeout read operations

#!/bin/bash

**if** [[ $# -le 0 ]]

**then**

printf "Not enough arguments!\n"

exit

**fi**

TIMEOUT=$1

VARIABLE=0

**while** :

**do**

((VARIABLE = VARIABLE + 1))

read -t $TIMEOUT -p "Do you want to Quit(Y/N): "

**if** [ $VARIABLE -gt $TIMEOUT ]; **then**

echo "Timing out - user response took too long!"

break

**fi**

**case** $REPLY in

[yY]\*)

echo "Quitting!"

break

;;

[nN]\*)

echo "Do not quit!"

;;

\*) echo "Please choose Y or N!"

;;

**esac**

**done**

### Converting tabs to spaces[Permalink](https://www.linode.com/docs/development/bash/solving-real-world-problems-with-bash-scripts-a-tutorial/#converting-tabs-to-spaces)

#!/bin/bash

**for** f in "$@"

**do**

**if** [ ! -f $f ]

**then**

echo $f does not exist!

**continue**

**fi**

echo "Converting $f.";

newFile=**$(**expand -t 4 "$f"**)**;

echo "$newFile" > "$f";

**done**

### Counting files

#!/bin/bash

DIRECTORIES="/bin:/home/mtsouk/code:/srv/www/www.mtsoukalos.eu/logs:/notThere"

# Count the number of arguments passed in

**if** [[ $# -le 0 ]]

**then**

echo "Using default value for COUNT!"

**else**

**if** [[ $1 =~ ^-?[0-9]+([0-9]+)?$ ]]

**then**

COUNT=$1

**fi**

**fi**

**while** read -d ':' dir; **do**

**if** [ ! -d "$dir" ]

**then**

echo "\*\*" Skipping $dir

**continue**

**fi**

files=`find $dir -type f | wc -l`

**if** [ $files -lt $COUNT ]

**then**

echo "Everything is fine in $dir: $files"

**else**

echo "WARNING: Large number of files in $dir: $files!"

**fi**

**done** <<< "$DIRECTORIES:"