A1: Write a script for printing all file related information in the present working directory (e.g.: size, permission & size etc…)

#!/bin/bash

ls -l $PWD

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A2: Read ‘n’ and generate a pattern given below

1

2 3

4 5 6

7 8 9 10

#!/bin/bash

read -p "enter the N value:" n

x=1

for((i=1; i<=$n; i++))

do

for((j=1; j<=$i; j++))

do

echo -n "$x "

((x++))

done

echo " "

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A3: Read ‘n’ and generate a pattern given below

1

1 2

1 2 3

1 2 3 4

#!/bin/bash

read -p "enter the N value:" n

for((i=1; i<=$n; i++))

do

for((j=1; j<=i; j++))

do

echo -n "$i"

done

echo " "

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A4: Write a script for scp & ssh by demonstrating remote login & file transfer functions:

echo "---------------------------------------SSH---------------------------------------"

while [ true ]

do

echo "1.Remote Login"

echo "2.Remote Copy"

echo "3.Exit"

read -p "Enter Your Choice [1/2/3]: " ch

case $ch in

1)

read -p "enter the UNAME@IP : " uname

ssh $uname

;;

2)

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

echo "make sure to add \"uname@ip:\" before the path of remote system:"

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

echo "enter the Absolute Path of file you want copy from: "

read src

echo "enter the Absolute Path of directory you want paste to: "

read dest

echo "src: $src"

echo "dest: $dest"

scp $src $dest

echo "Done.............."

;;

3) exit 0 ;;

\*) echo "!INVALID CHOICE" ;;

esac

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A5: Write a script for addition of two numbers for real numbers also.

read -p "enter your expression:: " val

echo $val | bc

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A6: Write a script for arithmetic calculator using command line arguments.

val=$@

echo $val | bc

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A7: Write a script to compare larger integer values from a ‘n’ number of arguments using command line arguments

#!/bin/bash

large=$1

echo "Total arguments: $#"

echo "ARGS: $\*"

for i in $\*

do

if [ $i -gt $large ]

then

large=$i

fi

done

echo "Large: $large"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A8: Write a script to print a given number in reverse order

#!/bin/bash

read -p "enter the number :" num

rev=0

while [ $num -gt 0 ]

do

l=$((num % 10))

rev=$((rev \* 10 + l))

num=$((num / 10))

done

echo "reverse of the number: $rev"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A9: Write a script to delete empty lines from a file

read -p "enter file name: " file

sed -i '/^$/ d' $file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A10: Write a script to perform arithmetic operation on digits of a given number depending upon the operator

read -p "enter the number: " num

read -p "enter the operator(+ - \* /): " op

echo "number: $num"

echo "operator: $op"

#read -p "enter number: " num

#echo "number: $num"

#len=${#num}

#echo "length: $len"

#for((i=0;i<$len;i++))

#do

#x=${num:$i:1}

#echo "$x"

#done

case $op in

+)

sum=0

len=${#num}

echo "length: $len"

for((i=0;i<$len;i++))

do

x=${num:$i:1}

sum=$(($x + $sum))

done

echo $sum

;;

-)

res=${num:0:1}

len=${#num}

echo "length: $len"

for((i=1;i<$len;i++))

do

x=${num:$i:1}

res=$(($res - $x))

done

echo "res: $res"

#temp=$num

#while [ $temp -gt 0 ]

#do

#rem=$(($temp % 10 ))

#echo "$rem"

#res=$(($res - $rem))

#temp=$(($temp / 10))

#done

#echo $res

;;

\\*)

#read -p "n1: " n1

#read -p "n2: " n2

#res=$(($n1 \* $n2))

#echo "res: $res"

sum=1

len=${#num}

echo "length: $len"

for((i=0;i<$len;i++))

do

x=${num:$i:1}

sum=$(($x \* $sum))

done

echo $sum

;;

/)

SCALE=2

res=${num:0:1}

len=${#num}

echo "length: $len"

for((i=1;i<$len;i++))

do

x=${num:$i:1}

if [ $x -eq 0 ]

then

continue

else

res=$(($res / $x))

fi

done

echo "res: $res"

;;

\*)

echo "invalid choice!!!!"

exit 0;

;;

esac

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A11: Read ‘n’ and generate Fibonacci numbers less than or equal to n

read -p "enter the range value for fibonacci series: " n

i=0

j=1

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

for((x=1; x<=n; x++))

do

k=$(($i + $j))

echo "$i"

i=$j

j=$k

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A12: Write a script to print the length of each and every string using arrays

#!/bin/bash

#read line

#echo "$line"

#for word in $line

#do

#echo "word-${#word} $word"

#done

arr=()

read -p "enter the line: " line

for word in $line

do

echo "$word"

arr+=("$word")

done

echo "inside array "

for w in ${arr[@]}

do

echo "$w -- ${#w}"

done

echo "array done"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A13: Write a script to print chess board.

#!/bin/bash

for((i=1; i<=8; i++))

do

for((j=1; j<=8; j++))

do

if [ $(( ($i + $j) % 2)) -eq 0 ]

then

echo -n "0 "

else

echo -n "1 "

fi

done

echo " "

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A14: Write a script to sort a given number in ascending or descending order.

#!/bin/bash

read -p " enter the numbers: " num

echo "numbers: $num"

echo "1.assending 2.descending "

read -p "enter your choice: " ch

if [ $ch == 1 ]

then

res=$(echo $num | tr " " "\n" | sort -n )

echo "$res"

elif [ $ch = 2 ]

then

res=$(echo $num | tr " " "\n" | sort -nr )

echo "$res"

else

echo "invalid option....."

fi

---------------------------------------------------------------------or---------------------------------------------------------------

arr=()

read -p "Enter the numbers to sort: " -a arr

len=${#arr[@]}

echo "$len"

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

echo "1. Ascending 2. Descending"

read -p "Select sorting order: " ch

if [ $ch -eq 1 ]; then

for ((i = 0; i < $len - 1; i++)); do

for ((j = 0; j < $len - 1 - i; j++)); do

if [ ${arr[j]} -gt ${arr[$((j + 1))]} ]; then

temp=${arr[j]}

arr[j]=${arr[$((j + 1))]}

arr[$((j + 1))]=$temp

fi

done

done

echo "Ascending sorted arr: ${arr[@]}"

else

for ((i = 0; i < $len - 1; i++)); do

for ((j = 0; j < $len - 1 - i; j++)); do

if [ ${arr[j]} -lt ${arr[$((j + 1))]} ]; then

temp=${arr[j]}

arr[j]=${arr[$((j + 1))]}

arr[$((j + 1))]=$temp

fi

done

done

echo "Descending sorted arr: ${arr[@]}"

fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A15: Write a script to print the following system information

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

echo "OS Information:"

echo " OS: $(uname -s)"

echo " Kernel: $(uname -r)"

echo " Architecture: $(uname -m)"

echo " Hostname: $(hostname)"

echo " Uptime: $(uptime -p)"

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

echo "Network Information:"

echo " IP Address: $(hostname -I | awk '{print $1}')"

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

echo "Logged in Users:"

who

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

echo "Memory Information:"

free -h

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A16: Read ‘n’ and print the greatest Fibonacci number less than or equal to n.

#!/bin/bash

read -p "Enter a number: " n

n1=0

n2=1

while true

do

nn=$((n1 + n2))

if [ $nn -gt $n ]; then

break

fi

n1=$n2

n2=$nn

done

echo "The greatest Fibonacci number less than or equal to $n is: $n2"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A17: Write a script to rename a file/directory replaced by lower/upper case letters.

read -p "enter the file name " file

if [[ -e $file || -d $file ]]

then

echo "1.conver to upper case"

echo "2.conver to lower case"

read -p "enter your choice:" ch

case $ch in

1)

if [ $file == ${file^^} ]

then

echo "already in uppercase: "

else

mv $file ${file^^}

fi

;;

2)

if [ $file == ${file,,} ]

then

echo "already in lowercase: "

else

mv $file ${file,,}

fi

;;

\*)echo "invalid choice!!!!"

exit 0 ;;

esac

else

echo "$file not exists"

fi

#echo "small: ${txt,,}"

#echo "large: ${txt^^}"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A18: Write a script to rename current working directory with given name.

#!/bin/bash

echo "cur PWD: $(pwd)"

read -p "enter new name for current directory: " new\_name

mv $(pwd) ../$new\_name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A19: Given album name and directory create a script to name files properly by inserting index numbers.

#!/bin/bash

read -p "Enter the directory name to be created: " dir

mkdir $dir

if [ $? -eq 0 ]; then

cd $dir

read -p "Enter the file name to be created: " file

read -p "Enter the number of files to be created: " tt

for ((i=1; i<=tt; i++)); do

touch ${file}\_$i

done

if [ $? -eq 0 ]; then

echo "Files created successfully"

else

echo "Files were not created"

fi

else

echo "Directory not created"

fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A20: Write script to print contents of file from given line number to next given number of lines.

#!/bin/bash

read -p "enter the file name: " file

if [ -e $file ]

then

read -p "enter the starting line number:" startline

read -p "enter the starting line number:" endingline

sed -n "${startline},${endingline} p" $file

else

echo "file not found!!!!!!!"

fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A21: Write script called say\_hello, which will print greetings based on time.

#!/bin/bash

current\_hour=$(date +%H)

echo "$current\_hour"

if [ "$current\_hour" -ge 5 ] && [ "$current\_hour" -lt 12 ]; then

echo "Good Morning!"

elif [ "$current\_hour" -ge 12 ] && [ "$current\_hour" -lt 18 ]; then

echo "Good Afternoon!"

elif [ "$current\_hour" -ge 18 ] && [ "$current\_hour" -lt 21 ]; then

echo "Good Evening!"

else

echo "Good Night!"

fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A22: Shell script to convert string lower to upper and upper to lower.

#!/bin/bash

#read -p "Enter a string: " input\_string

#upper\_case="${input\_string^^}"

#lower\_case="${input\_string,,}"

#echo "Original String: $input\_string"

#echo "Converted to Uppercase: $upper\_case"

#echo "Converted to Lowercase: $lower\_case"

read -p "Enter a string: " input\_string

if [[ "$input\_string" == "${input\_string,,}" ]]; then

echo "Input is in lowercase, converting to uppercase..."

converted\_string="${input\_string^^}"

elif [[ "$input\_string" == "${input\_string^^}" ]]; then

echo "Input is in uppercase, converting to lowercase..."

converted\_string="${input\_string,,}"

else

echo "Input contains both uppercase and lowercase characters, converting to uppercase..."

converted\_string="${input\_string^^}"

fi

echo "Original String: $input\_string"

echo "Converted String: $converted\_string"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A23: Display the longest and shortest user-names on the system.

#!/bin/bash

usernames=$(cut -d: -f1 /etc/passwd)

longest=""

shortest=""

for username in $usernames; do

if [ -z "$shortest" ] || [ ${#username} -lt ${#shortest} ]; then

shortest=$username

fi

if [ ${#username} -gt ${#longest} ]; then

longest=$username

fi

done

echo "Shortest username: $shortest"

echo "Longest username: $longest"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A24: Write a script to delete all the .swp files found in your system or directory.

search\_dir="."

find "$search\_dir" -type f -name "\*.swp" -exec rm -f {} \;

echo "All .swp files delete."

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A25: Write a script for generating random 8-character passwords including alpha numeric characters.

#!/bin/bash

password=$(</dev/urandom tr -dc 'A-Za-z0-9' | head -c 8)

echo "Generated password: $password"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A26: Write a script that takes any number of directories as command-line arguments and then lists the contents of each of these directories.

#!/bin/bash

if [ $# -eq 0 ]; then

echo "Usage: $0 <directory1> <directory2> ... <directoryN>"

exit 1

fi

for dir in "$@"; do

if [ -d "$dir" ]; then

echo "Listing contents of directory: $dir"

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

ls "$dir" # List the contents of the directory

echo ""

else

echo "Error: $dir is not a directory!"

echo ""

fi

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A27: Use pipes or redirection to create an infinite feedback loop.

#cat | cat

#cat <&0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A28: Use a recursive function to print each argument passed to the function.

#!/bin/bash

print\_args() {

if [ $# -eq 0 ]; then

return

fi

echo "$1"

print\_args "${@:2}"

}

print\_args "$@"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A29: Write a script to determine whether a given file system or mount point is mounted.

read -p "Enter the mount point or file system to check (e.g / or /dev/sda1): " mount\_point

if [ -z "$mount\_point" ]; then

echo "You must enter a valid mount point."

exit 1

fi

if grep -qs "$mount\_point" /proc/mounts; then

echo "The mount point '$mount\_point' is mounted."

else

echo "The mount point '$mount\_point' is not mounted."

fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A30: Write a script to locks down file permissions for a particular directory.

read -p "Enter the directory path to lockdown: " dir

if [ ! -d "$dir" ]; then

echo "Directory '$dir' does not exist!"

exit 1

fi

chmod 700 "$dir"

find "$dir" -type d -exec chmod 700 {} \;

find "$dir" -type f -exec chmod 600 {} \;

echo "Permissions for '$dir' and its contents have been locked down."

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A31: Display the names of any file-system which have less than 10% free space available.

Df –h | awk ‘{ if ($5+0 > 90) print $1 }’

To create large file:

dd if=/dev/zero of=filex bs=1M count=10240

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A32: Count the number of users with user IDs between 500 and 10000 on the system.

#!/bin/bash

count=0

while IFS=: read -r username password uid gid full\_name home shell; do

if [ "$uid" -ge 500 ] && [ "$uid" -le 10000 ]; then

((count++))

fi

done < /etc/passwd

echo "Number of users with UID between 500 and 10000: $count"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A33: For each directory in the $PATH, display the number of executable files in that directory.

#!/bin/bash

IFS=':' read -ra dirs <<< "$PATH"

for dir in "${dirs[@]}"; do

if [ -d "$dir" ]; then

count=$(find "$dir" -maxdepth 1 -type f -executable | wc -l)

echo "Directory: $dir - Executable files: $count"

else

echo "Directory: $dir does not exist."

fi

done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A34: Write a script to search a user present in your system.

read -p "enter user name to be searched: " uname

names=($(awk -F: '{print $1}' /etc/passwd))

for name in ${names[@]}

do

if [ $name == $uname ]

then

echo "user exists............."

exit 0

fi

done

echo "user not present in system....... "

#if (grep -w "$uname" /etc/passwd )

#then

#echo "user present.............."

#else

#echo "user not present !!!!!!!!11"

#fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A35: Write a script to replace 20% lines in a C file randomly and replace it with the pattern

file="f1"

total\_lines=$(cat $file | wc -l)

echo "total lines: $total\_lines"

lines\_to\_modify=$(($total\_lines \* 20 / 100))

echo "linesto modify: $lines\_to\_modify "

for((i=0;i<$lines\_to\_modify;i++))

do

x=$((RANDOM % $total\_lines))

random\_val=$(echo "------------------$RANDOM $RANDOM $RANDOM $RANDOM $RANDOM----------------- ")

echo "$x"

sed -i "$x s/.\*/${random\_val}/" "$file"

done