|  |  |  |
| --- | --- | --- |
| INDEX | | |
| SL.NO | PROGRAMS | PAGE NO. |
| 1 | Write a shell script that takes a valid directory name as an argument recursively descend all the sub-directors, find the maximum length of any file in that hierarchy, and write the maximum value to the standard output. | 1 |
| 2 | Write a shell script that accepts a path name and creates all the components in that path name as directories. For example, if the script is named as mpc, then the command mpc a/b/c/d should create subdirectories a, a/b, a/b/c, a/b/c/d. | 2-3 |
| 3 | Write a shell script that accepts two filenames as arguments, checks if the permissions for these files are identical and if the permissions are identical, output common permissions otherwise output each filename followed by its permissions. | 4-5 |
| 4 | Write a shell script which accepts valid log-in names as arguments and prints their corresponding home directories, if no arguments are specified, print a suitable error message. | 6 |
| 5 | Create a script file called file properties that reads a filename entered and outputs its properties. | 7-8 |
| 6 | Create a script file called file properties that reads a filename entered and outputs its properties. | 9-10 |
| 7 | Write a shell script to implement terminal locking (Similar to the lock command). It should prompt for the user for a password. After accepting the password entered by the user, it must prompt again for the matching password as confirmation and if match occurs, it must lock the keyword until a matching password is entered again by the user. | 11 |
| 8 | Write a shell script that accept one or more file names as argument and convert all of them to uppercase, provided they exists in current directory. | 12-13 |
| 9 | Write a shell script that displays all the links to a file specified as the first argument to the script. The second argument, which is optional, can be used to specify in which the search is to begin. If this second argument is not present, the search is to begin in the current working directory. In either case, the starting directory as well as its subdirectories at all levels must be searched. The script need not include error checking. | 14 |
| 10 | Write a shell script that accepts filename as argument and display its creation time if file exist and if does not send output error message. | 15 |
| 11 | Write a shell script to display the calendar for the current month with current date replaced by \* or \*\* depending whether the date is one digit or two digit. | 16-17 |
| 12 | Write a shell script to find a file/s that matches a pattern given as command line argument in the home directory, display the contents of the file and copy the file into the directory ~/mydir. | 19 |
| 13 | Write an awk script that accepts date argument in the form of dd-mmyy and display it in the form month, day and year. The script should check the validity of the argument and in the case of error, display a suitable message. | 20-21 |
| 14 | Write an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged. | 22 |



# 

1.Write a shell script that takes a valid directory name as an argument recursively descend all the sub-directors, find the maximum length of any file in that hierarchy, and write the maximum value to the standard output.

# AIM

Script to take a valid directory name as an argument recursively descend all the sub-directors, find the maximum length of any file in that hierarchy, and write the maximum value to the standard output.

# SOURCE CODE

# #!/bin/sh

# for i in $\*

# do

# if [ -d $i ]

# then

# echo " large file size is "

# else

# echo " not a directory "

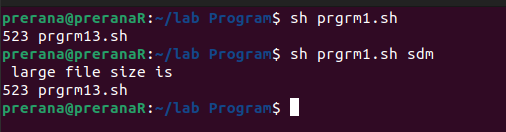
# fi

# done

# echo `ls -Rl $1 | grep "^-" | tr -s ' ' | cut -d ' ' -f 5,9 | sort -n |

# tail -1`

# OUTPUT



# 

2. Write a shell script that accepts a path name and creates all the components in that path name as directories. For example, if the script is named as mpc, then the command mpc a/b/c/d should create subdirectories a, a/b, a/b/c, a/b/c/d.

# AIM

Shell script to accepts a path name and creates all the components in that path name as directories. For example, if the script is named as mpc, then the command mpc a/b/c/d should create subdirectories a, a/b, a/b/c, a/b/c/d.

# SOURCE CODE

#!/bin/bash

echo " enter the pathname"

read p

i=1

j=1

len=`

echo $p | wc -c`

while [ $i -le $len ]

do

x=`echo $p | cut -d ' ' -f $j`

namelength=`echo $x | wc -c`

mkdir -p $x

cd $x

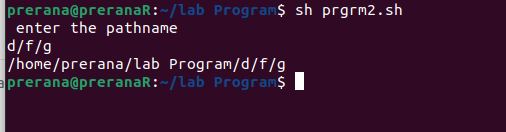
pwd

j=`expr $j + 1`

i=`expr $i + $namelength`

done

# OUTPUT



3.Write a shell script that accepts two filenames as arguments, checks if the permissions for these files are identical and if the permissions are identical, output common permissions otherwise output each filename followed by its permissions.

# AIM

Shell script to accepts two filenames as arguments, checks if the permissions for these files are identical and if the permissions are identical, output common permissions otherwise output each filename followed by its permissions.

# SOURCE CODE

#!/bin/sh

echo "Enter file name 1 :$1"

echo "Enter file name 2 :$2"

if [ $# -eq 0 ]

then

echo "no arguments passed"

elif [ ! -e $1 -o ! -e $2 ]

then

echo "file does not exist"

else

x=`ls -l $1 | cut -c 1-10`

y=`ls -l $2 | cut -c 1-10`

if [ $x = $y ]

then

echo "permissions are same : $x"

else

echo "permissions are different"

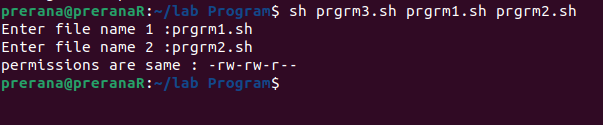
echo "permission of $1 is $x"

echo "permission of $2 is $y"

fi

fi

# OUTPUT



4.Write a shell script which accepts valid log-in names as arguments and prints their corresponding home directories, if no arguments are specified, print a suitable error message.

# AIM

Shell script to accepts valid log-in names as arguments and prints their corresponding home directories, if no arguments are specified, print a suitable error message.

SOURCE CODE

# #!/bin/sh

# echo "enter arguments : $1";

# if [ $# -eq 0 ]

# then

# echo "No command line argument passed"

# exit

# fi

# while [ $1 ]

# do

# cat /etc/passwd | cut -d ":" -f1 | grep "^$1" > temp

# a=`cat temp`

# if [ "$a" != "$1" ]

# then

# echo "ERROR:$1 is an invalid login name"

# else

# echo "Home Directory for $1 is"

# echo `cat /etc/passwd | grep "^$1" | cut -d ":" -f6`

# fi

# shift

# done

# OUTPUT

# 

5.Create a script file called file properties that reads a filename entered and outputs its properties.

AIM

Shell script to reads a filename entered and outputs its properties.

# SOURCE CODE

#!/bin/bash

echo "enter the file name"

read file

if [ -f $file ]

then

set -- `ls -l $file`

echo "file permission $1"

echo "number of links $2"

echo "user name $3"

echo "group name $4"

echo "file size $5 bytes"

echo "date of modification $6 $7"

echo "time of modification $8"

echo "name of file $9"

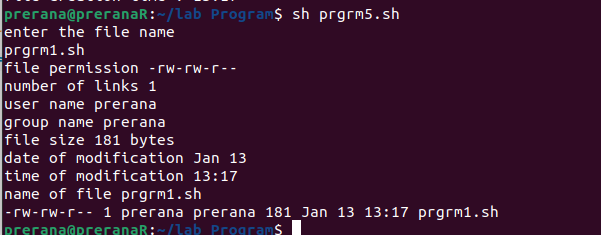
else

echo "file does not exit"

fi

ls -l $file

OUTPUT:



6.Write a shell script to implement terminal locking (Similar to the lock command). It should prompt for the user for a password. After accepting the password entered by the user, it must prompt again for the matching password as confirmation and if match occurs, it must lock the keyword until a matching password is entered again by the user.

# AIM

Shell script to implement terminal locking (Similar to the lock command). It should prompt for the user for a password. After accepting the password entered by the user, it must prompt again for the matching password as confirmation and if match occurs, it must lock the keyword until a matching password is entered again by the user.

# SOURCE CODE

#!/bin/bash

stty -echo

clear

echo "Enter the password"

read pass1

echo "Re-Enter the password"

read pass2

val=1

while [ $val -eq 1 ]

do

if [ $pass1 = $pass2 ]

then

val=0

echo "Terminal locked"

echo "To unlock enter the password"

pass1=""

until [ "$pass1" = "$pass2" ]

do

read pass1

echo "Terminal locked

done

echo "Terminal unlocked"

stty echo

else

echo "Password mismatch please re-type it"

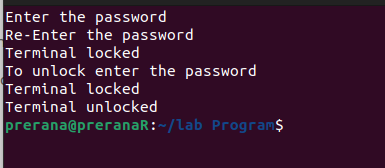
stty -echo

read pass2

fi

done

# OUTPUT



7.Write a shell script that accept one or more file names as argument and convert all of them to uppercase, provided they exists in current directory.

# AIM

Shell script to accept one or more file names as argument and convert all of them to uppercase, provided they exists in current directory.

# SOURCE CODE

#!/bin/bash

echo "Enter the file name"

read file

if [ -z $file ]

then

echo "no arguments passed"

elif [ ! -f $file ]

then

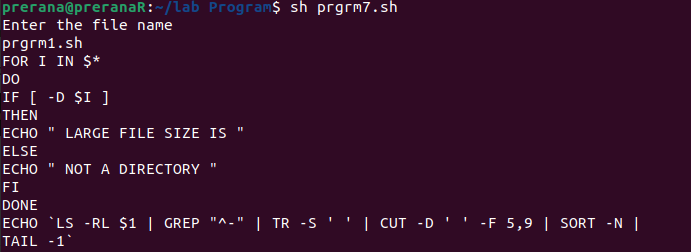
echo "file does not exist"

else

tr '[a-z]' '[A-Z]' < $file

fi

OUTPUT:



8.Write a shell script that displays all the links to a file specified as the first argument to the script. The second argument, which is optional, can be used to specify in which the search is to begin. If this second argument is not present, the search is to begin in the current working directory. In either case, the starting directory as well as its subdirectories at all levels must be searched. The script need not include error checking.

# AIM

Shell script to displays all the links to a file specified as the first argument to the script. The second argument, which is optional, can be used to specify in which the search is to begin. If this second argument is not present, the search is to begin in the current working directory. In either case, the starting directory as well as its subdirectories at all levels must be searched.

The script need not include error checking.

# SOURCE CODE

#!/bin/bash

file=$1

set -- `ls -l $file`

lcnt=$2

if [ $lcnt -eq 1 ]

then

echo "no other links"

exit

else

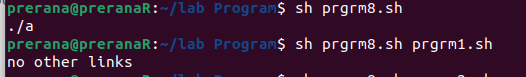
set -- `ls -i $file`

inode=$1

find "." -xdev -inum $inode -print

fi

OUTPUT:



9.Write a shell script that accepts filename as argument and display its creation time if file exist and if does not send output error message.

# AIM

Shell script to accepts filename as argument and display its creation time if file exist and if does not send output error message.

# SOURCE CODE

#!/bin/bash

echo "Enter the file name"

read file

if [ -z $file ]

then

echo "no arguments passed"

elif [ ! -f $file ]

then

echo "file does not exist"

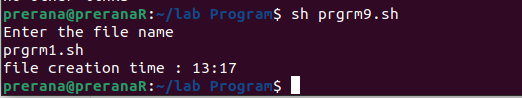
else

f=`ls -l $file | cut -d " " -f8`

echo "file creation time :" $f

fi

OUTPUT:



10.Write a shell script to display the calendar for the current month with current date replaced by \* or \*\* depending whether the date is one digit or two digit.

# AIM

Shell script to display the calendar for the current month with current date replaced by \* or \*\* depending whether the date is one digit or two digit.

# SOURCE CODE

#!/bin/bash

set `date`

y=$2

if [ $y -le 9 ]

then

date |sed "s/$2/\*/"

else

date |sed "s/$2/\*\*/"

fi

OUTPUT:



11.Write a shell script to find a file/s that matches a pattern given as command line argument in the home directory, display the contents of the file and copy the file into the directory ~/mydir.

# AIM

Shell script to find a file/s that matches a pattern given as command line argument in the home directory, display the contents of the file and copy the file into the directory ~/mydir.

# SOURCE CODE

#!/bin/sh

echo "Enter the file name"

read file

if [ -z $file ]

then

echo "no arguments passed"

elif [ ! -f $file ]

then

echo "file does not exist"

else

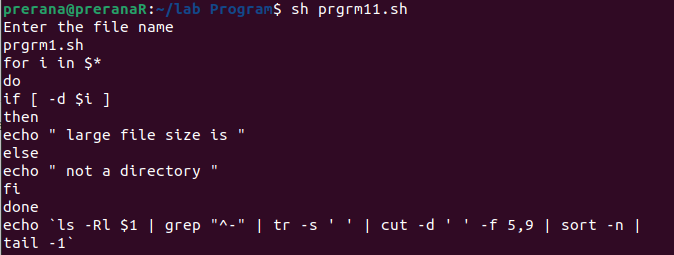
#ls $file

cat $file

cp -f $file /home/prerana/mydir

fi

OUTPUT:



12.Write a shell script to list all the files in a directory whose filename is at least 10 characters. (Use expr command to check the length).

# AIM

Shell script to list all the files in a directory whose filename is at least 10 characters. (Use expr command to check the length).

# SOURCE CODE

#!/bin/bash

if [ $# -eq 0 ]

then

echo "no argument"

else

c=`ls $1`

echo "filename are\n$c"

for i in $c

do

len=`expr length $i`

if [ $len -ge 10 ]

then

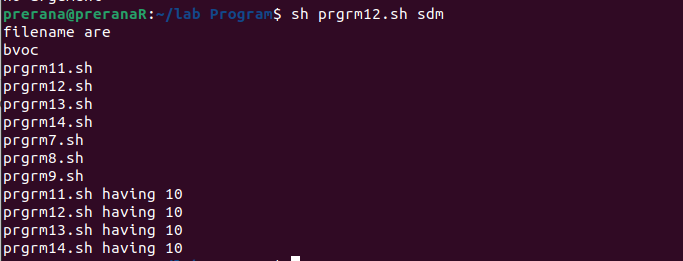
echo "$i having $len"

fi

done

fi

OUTPUT:



13.Write an awk script that accepts date argument in the form of dd-mmyy and display it in the form month, day and year. The script should check the validity of the argument and in the case of error, display a suitable message.

# AIM

Awk script to accepts date argument in the form of dd-mm-yy and display it in the form month, day and year. The script should check the validity of the argument and in the case of error, display a suitable message.

SOURCE CODE

awk '{ split ($0, arr, "-")

if ((arr[2] < 1) || (arr[2] > 12) || (arr[1] < 1) || (arr[1] > 31))

{

print "invalid date"}

else

{

if (arr[2] == 1)

print "jan"

if (arr[2] == 2)

print "feb"

if (arr[2] == 3)

print "march"

if (arr[2] == 4)

print "apirl"

if (arr[2] == 5)

print "may"

if (arr[2] == 6)

print "jun"

if (arr[2] == 7)

print "july"

if (arr[2] == 8)

print "aug"

if (arr[2] == 9)

print "sept"

if (arr[2] == 10)

print "oct"

if (arr[2] == 11)

print "nov"

if (arr[2] == 12)

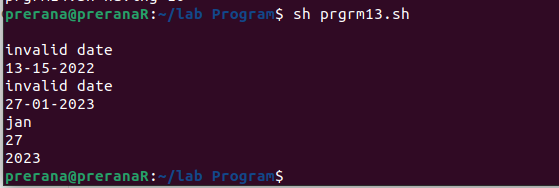
print "dec"

print arr[1]

print arr[3]

exit 0 } }'

OUTPUT:



14.Write an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.

# AIM

Awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.

# SOURCE CODE

#!/bin/sh

echo "Enter file name"

read file

if [ -z $file ]

then

echo “no arguments”

elif [ ! -f $file ]

then

echo “files does not exist”

else

awk '!visited[$0]++' $file

fi

OUTPUT:

