#1. Write a bash script that calculates the sum of the sizes (in bytes) of all regular files in a folder given as a parameter.(use test to check if the folder exists and if a given file is a regular file)

F=$1

if [ -z "$1" ]; then

echo No parameter was given

exit 1

fi

if [ ! -d "$1" ]; then

echo Given folder does not exist

exit 1

fi

sum=0

for item in $(ls "$1"); do

f="$1/$item"

if [ -f "$f" ]; then

size=`du -bs "$f" | awk '{print $1}'`

sum=`expr $sum + $size `

fi

done

#2. Write a script that reads filenames until the word "stop" is entered. For each filename, check if it is a text file and if it is, print the number of words on the first line.(Hint: test command to check if regular file; file command to check if text file)

while true; do

read input

if [ "$input" = "stop" ]; then

echo Done!

break

fi

if [ ! -f "$input" ]; then

echo Not a file!

fi

extension=`echo "$input" | cut -d "." -f2`

if [ "$extension" = "txt" ]; then

nr\_word=`cat "$input" | head -1 | wc -w`

echo the file "$input" has "$nr\_word" word on th first row

else

echo Not a text file!

fi

done

#!/bin/bash

#3. Write a script that receives as command line arguments pairs consisting of a filename and a word. For each pair, check if the given word appears at least 3 times in the file and print a corresponding message.

while true; do

if [ $# -eq 0 ]; then

echo No parameters were given!

break

fi

file=$1

word=$2

if [ ! -f "$file" ]; then

echo Not a file!

else

nr\_words=`cat "$file" | grep -E "$word" | wc -w`

if [ "$nr\_words" -eq 3 ]; then

echo The word "$word" was found 3 times in the file "$file"

else

echo The word "$word" was not found 3 times in the file "$file"

fi

fi

if [ $# -ge 4 ]; then

shift 2

else

echo Done!

break

fi

done

#!/bin/bash

#4. Write a bash script that sorts all files given as command line arguments descending by size.(firstcheck if an argument is a file)

echo "" > out.txt

for file in $@; do

if [ -f "$file" ]; then

echo "$file" `du -b "$file"` >> out.txt

fi

done

#!/bin/bash

#5. Write a script that extracts from all the C source files given as command line arguments the included libraries and saves them in a file.(use the file command to check if a file is a C source file)

if [ $# -eq 0 ]; then

echo No parameters were given!

break

fi

for file in $@; do

if file "$file" | grep -q "C program"; then

echo `grep -E "^#include" "$file"` >> out\_ex\_5.txt

fi

done

#!/bin/bash

#6. Write a script that monitors the state of a given folder and prints a message when something changed.

if [ $# -eq 0 ];then

echo "No parameter was given!"

exit 1

fi

folder=$1

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

echo "Folder does not exist!"

exit 1

fi

state=""

while true; do

s=""

for child in `find "$folder"`; do

if [ -f "$child" ]; then

LS=`ls -l "$child" | sha1sum`

CONTENT=`sha1sum $child`

elif [ -d "$child" ]; then

LS=`ls -l -d "$child" | sha1sum`

CONTENT=`ls -l "$child" | sha1sum`

fi

S="$S\n$LS $CONTENT"

done

if [ -n "$state" ] && [ "$S" != "$state" ]; then

echo "Folder state has changed!"

else

echo "Folder state hasn't changed!"

fi

state="$S"

sleep 1

done

#!/bin/bash

#7. Find recursively in a given directory all the symbolic links, and report those that point to files/directories that no longer exist. Use option -L to test if a path is a symbolic link, and option -e to test if it exists (will return false if the target to which the link points does not exist)

if [ $# -eq 0 ];then

echo "No paramters was given!"

exit 1

fi

directory=$1

if [ ! -d "$1" ]; then

echo "Directory does not exist!"

exit 1

fi

for child in `find "$directory" -type l`; do

if [ ! -e "$child" ]; then

echo "Link "$child" is not a valid one!"

fi

done

#!/bin/bash

#8. Write a bash script that receives a folder name as argument. Find recursively in the folder the number of times each file name is repeated.

if [ $# -eq 0 ];then

echo "No parameter was given!"

exit 1

fi

directory=$1

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

echo "The given directory does not exist!"

exit 1

fi

find "$directory" | awk 'NR != 1' | awk -F"/" '{print $2}' | sort | uniq -c

#!/bin/bash

#Write a shell script that receives a number and however many directory names. Display all the regular files from each directory (recursive) that have their size in bytes grater than the given number

if [ "$#" -eq 0 ];then

echo "No parameters were given!"

exit 1

fi

if [ ! -d "$2" ]; then

echo "No directory was given!"

exit 1

fi

size=$1

shift 1

while true; do

if [ ! -d "$1" ]; then

echo "Given directory does not exist!"

exit 1

fi

directory=$1

for child in `find "$directory" -type f`;do

if [ `du -b "$child" | awk '{print $1}'` -gt "$size" ];then

echo "File: `ls "$child" | awk -F"/" '{print $2}'`"

fi

done

if [ "$#" -ge 2 ];then

shift 1

else

echo "Done!"

break

fi

done

#!/bin/bash

#9. Calculate the average of all process ids in the system per user.

list\_of\_processes=`ps -ef | grep -E "^[a-z]{4}[0123456789]{4}" | awk 'NR > 1{print $1","$2}' | sort`

p=`ps -ef | grep -E "^[a-z]{4}[0123456789]{4}" | awk '{print $1","$2}' | sort | head -1`

p1\_user=`echo "$p1" | cut -d"," -f1`

p1\_id=`echo "$p1" | cut -d"," -f2`

sum=$((sum+p1\_id))

nr\_of\_processes=1

for p in $list\_of\_processes; do

p2\_user=`echo "$p" | cut -d"," -f1`

p2\_id=`echo "$p" | cut -d"," -f2`

if [ "$p1\_user" == "$p2\_user" ]; then

nr\_of\_processes=$((nr\_of\_processes+1))

sum=$((sum+p2\_id))

else

if [ $nr\_of\_processes -gt 0 ]; then

echo "Average for "$p1\_user" is "$((sum/nr\_of\_processes))

fi

p1\_user="$p2\_user"

sum=0

nr\_of\_processe=0

fi

done

#!/bin/bash

#10. Write a script that receives program/process names as command line arguments. The script will monitor all the processes in the system, and whenever a program with one of those names is run, the script will kill it and display a message. (see commands ps, kill, killall).Alternatively, use pgrep/pkill

if [ "$#" -eq 0 ]; then

echo "No parameters were given"

exit 1

fi

while true; do

for process in $@; do

process\_id=""

process\_id=`ps -ef | awk '{print $8" "$2}' | grep -E "/<$process " | awk '{print $2}'`

if [ -n "$process\_id" ]; then

kill -9 "$process\_id"

fi

done

done

#!/bin/bash

#11. Write a script that receives a directory as a command line argument. The script will delete all the C source files from the directory and will display all other text files sorted alphabetically.

if [ $# -ne 1 ]; then

echo "Invalid number of parameters"

exit 1

fi

directory=$1

for file in `find "$directory" -type f -name "\*.c"`; do

`rm "$file"`

done

for file in `find "$directory" | sort`; do

done

#!/bin/bash

#12. Write a script that finds recursively in the current folder and displays all the regular files that have write permisions for everybody (owner, group, other). Then the script removes the write permissions from everybody. Hint: use chmod's symbolic permissions mode (see the manual).

current\_dir="."

for file in `find "$current\_dir" -type f`; do

permisison=""

permission=`ls -all | awk '{print "$1"}' | grep -E "w{3}"`

if [ -n "$permission"]; then

echo "$file"

fi

`chmod o-w "$file"`

done

#!/bin/bash

#13. Consider a file containing a username on each line. Generate a comma-separated string with email addresses of the users that exist. The email address will be obtained by appending "@scs.ubbcluj.ro" at the end of each username. Make sure the generated string does NOT end in a comma.

if [ $# -eq 0 ]; then

echo "No parameters were given"

exit 1

fi

if [ ! -f $1 ]; then

echo "Given parameter is not a file"

exit 1

fi

result=""

for name in `cat $1`; do

if [ $result != "" ]; then

result="$result,$name@scs.ubbcluj.ro"

else

result="$name@scs.ubbcluj.ro"

fi

done

result=`echo $result | sed -E "s/,$//"`

echo $result

#!/bin/bash

#14. Write a shell script that recieves any number of words as command line arguments, and continuously reads from the keyboard one file name at a time. The program ends when all words received as parameters have been found at least once across the given files.

declare -A words

for i in $@; do

words[$i]=0

done

found\_all=false

while ! $found\_all; do

found\_all=true

read -p "Input a file a name: " file

if [ -z "$file" ]; then

echo "No file was given"

elif [ ! -f "$file" ]; then

echo "Given parameter is not a file"

else

for word in ${!words[@]}; do

if [ -n "`grep -E "\<$word\>" "$file"`" ]; then

words[$word]=1

fi

if [ ${words[$word]} -eq 0 ]; then

found\_all=false

fi

done

fi

done

echo "Done!"

#!/bin/bash

#15. Write a shell script that, for all the users in /etc/passwd, creates a file with the same name as the username and writes in it all the ip addresses from which that user has logged in. (hint: use the last command to find the ip addresses)

for line in `grep -E "^[a-z]{4}[0123456789]{4}" /etc/passwd | sed "s/,/:/gi" | sed "s/ /-/gi"`; do

username="`echo $line | cut -d":" -f 1`"

name="`echo $line | awk -F":" '{print $5}'`"

if [ -n "`last | grep -E "^"$username""`" ]; then

echo "`last | grep -E "^"$username"" | awk '{print $3}'`" >>"$name"

fi

done

#!/bin/bash

#trytest2. Read from the command line an argument (a directory) and search in it recursively. print the .sh files and print the total number of lines i the other files

if [ $# -ne 1 ]; then

echo "Invalid number of parameters"

exit 1

fi

if [ ! -d "$1" ]; then

echo "The provided parameter is not a directory"

exit 1

fi

dir=$1

for file in `find "$dir" -type f`; do

if [ -n "`echo "$file" | grep -E "\.sh\>"`" ]; then

echo "File: "$file""

else

echo "File: "$file" has "`cat "$file" | awk 'END{print NR}'`" lines"

#or use echo "File: "$file" has "`cat "$file" | wc -l`" lines"

fi

done

#!/bin/bash

#ex1 Boian. Display a report showing the full name of all the users currently connected, and the number of processes belonging to each of them

for user in `who | awk '{print $1}' | uniq`; do

name=`cat /etc/passwd | grep -E "^"$user"" | sed "s/,/:/gi" | awk -F":" '{print $5}'`

nr\_of\_processes=`ps -ef | grep -E "^"$user"" | awk '{print $1}' | uniq -c | awk '{print $1}'`

echo "the user "$name" has "$nr\_of\_processes" processes currently running"

done

read line by line

while IFS= read -r line

do

if [ $i -eq 0 ]

then

stringfinal=${stringfinal}$line$mail

else

stringfinal=${stringfinal}", "$line$mail

fi

i=`expr i + 1`

done < "$input"

#!/bin/bash

if [ $# -ne 1 ]; then

echo "Invalid number of parameters"

exit 1

fi

if [ ! -d "$1" ]; then

echo "Given parameter is not a directory"

exit 1

fi

dir=$1

for file in `find "$dir" -type f`; do

if [ -z "`echo "$file" | grep -E "\.c\>"`" ]; then

echo "File: "$file""

else

echo "C file ("$file") content: "

echo "`grep -E -v "#include" "$file"`"

fi

done