

Universidad de Investigación de Tecnología
Experimental Yachay Tech
School of Mathematics and Computer sciences

Program : Undergraduate _____ Semester : VII
Subject : HPC Professor : Saravana Prakash T
Date : 11/02/2019 Time:
Semester: Jan-May-2019
Student's Surname, Name : **Román Eras, Osiris Anael**
Student Reg.No:

Duration: 40 minutes

Take-home/Quiz/exercises

Maximum Marks: 100

Neatly show all of your work. If you need to use extra paper, please indicate so and attach to this exam.

Mode of submission : Upload your answers in D2L or hard-copy at my desk

Turn down the quiz /submit : Monday (11/02/2019), at 1:00 PM.

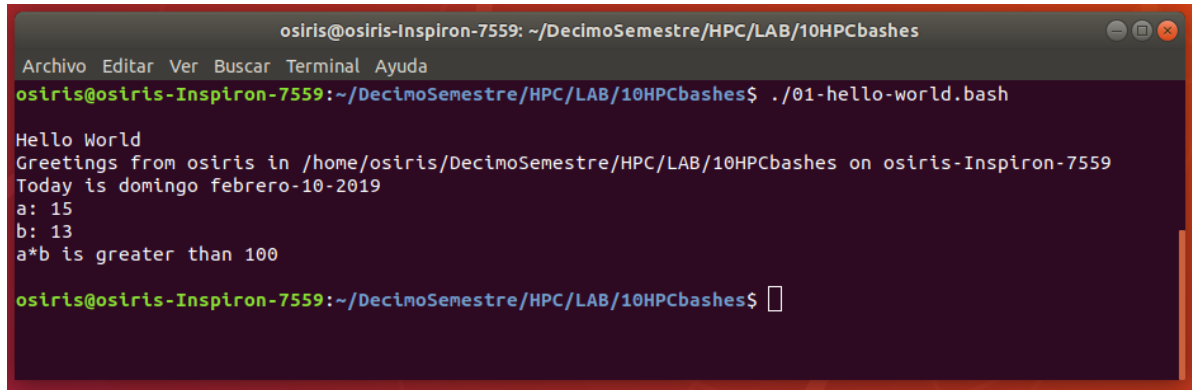
My Lab work

1. The next script prints Hello World, the path from where the script is executed, the laptop name, the date and hour and also tells if the result of a multiplication is greater or not than 100.

Listing 1: bash version: Example 1

```
1 #!/bin/bash
2 ## Print Hello world ...
3 echo
4
5 echo "Hello World"
6
7 echo "Greetings from" $USER "in" `pwd` "on" $HOSTNAME
8
9 echo "Today is" `date +"%A %B-%d-%Y" `
10
11 a=15;
12 b=13;
13 c=100;
14
15 echo "a: ${a} "
16 echo "b: ${b}"
17
18 if [ $((a*b)) -lt $c ]; then
19     echo "a*b is less than 100"
20 else
21
```

```
22     echo "a*b is greater than 100"
23 fi
24
25 echo
```

A screenshot of a terminal window titled "osiris@osiris-Inspiron-7559: ~/DecimoSemestre/HPC/LAB/10HPCbashes". The terminal shows the execution of a script named "01-hello-world.bash". The output of the script is displayed as follows:

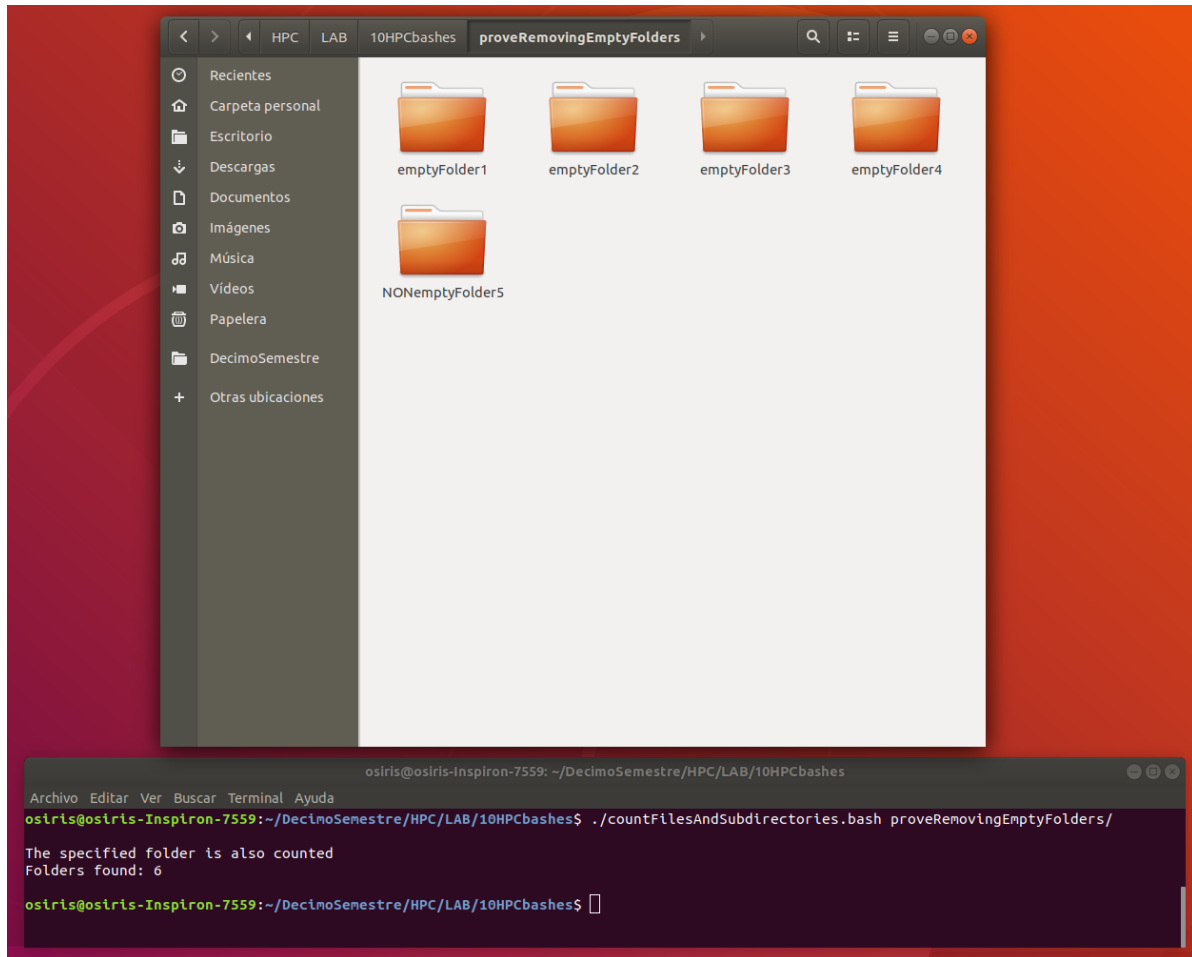
```
osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$ ./01-hello-world.bash
Hello World
Greetings from osiris in /home/osiris/DecimoSemestre/HPC/LAB/10HPCbashes on osiris-Inspiron-7559
Today is domingo febrero-10-2019
a: 15
b: 13
a*b is greater than 100
osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$
```

Executing in Terminal: OUTPUT

2. The next script takes a path or a directory name and recursively counts all the sub directories, the specified folder is also counted.

Listing 2: bash version: Example 2

```
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 echo
5
6 echo "The specified folder is also counted"
7
8 if [ -d "$@" ]; then
9 #The next line prints also the number of files existent
10 #     echo "Files found: $(find "$@" -type f | wc -l)"
11     echo "Folders found: $(find "$@" -type d | wc -l)"
12 else
13     echo "[ERROR] Please retry with a folder."
14     exit 1
15 fi
16 echo
```



Executing in Terminal: OUTPUT

3. The next script takes a path or a directory name and print the names of all empty sub directories into an output file.

Listing 3: bash version: Example 3

```

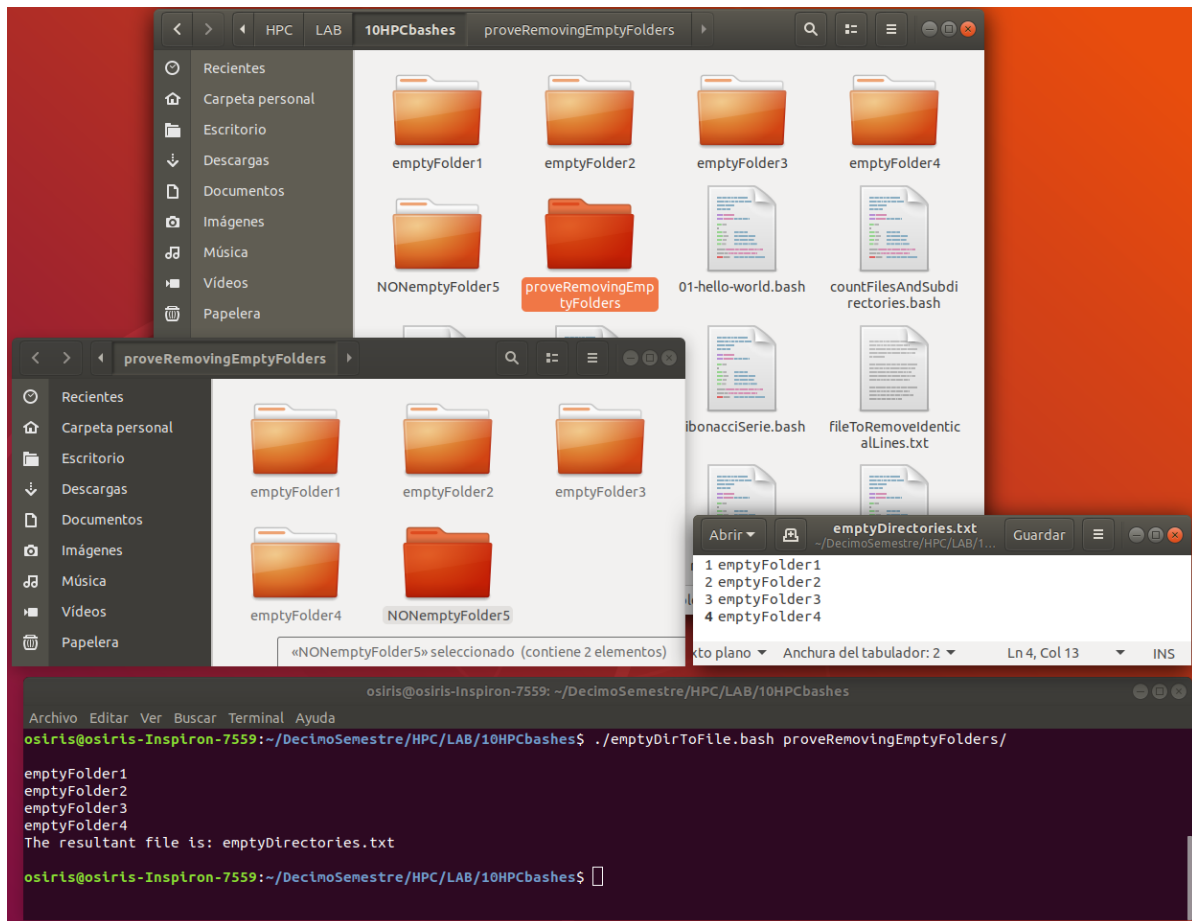
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 echo
5 folder=$1
6 file_out=emptyDirectories.txt
7
8 rm -f ${file_out}
9
10 LIST=$(ls ${folder})
11
12 for rdir in ${LIST}
13 do

```

```

14     if [ -n "$(find "$rdir" -maxdepth 0 -type d -empty 2>/dev/null)" ]; then
15         echo "${rdir}"
16         echo "${rdir}" >> ${file_out}
17     fi
18 done
19
20 echo "The resultant file is: ${file_out}"
21 echo

```



Executing in Terminal: OUTPUT

- The next script display a list of Fibonacci numbers, the number of elements displayed is defined by the user at the beginning, if the user doesn't specify a number, the first 10 elements of the Fibonacci series are displayed.

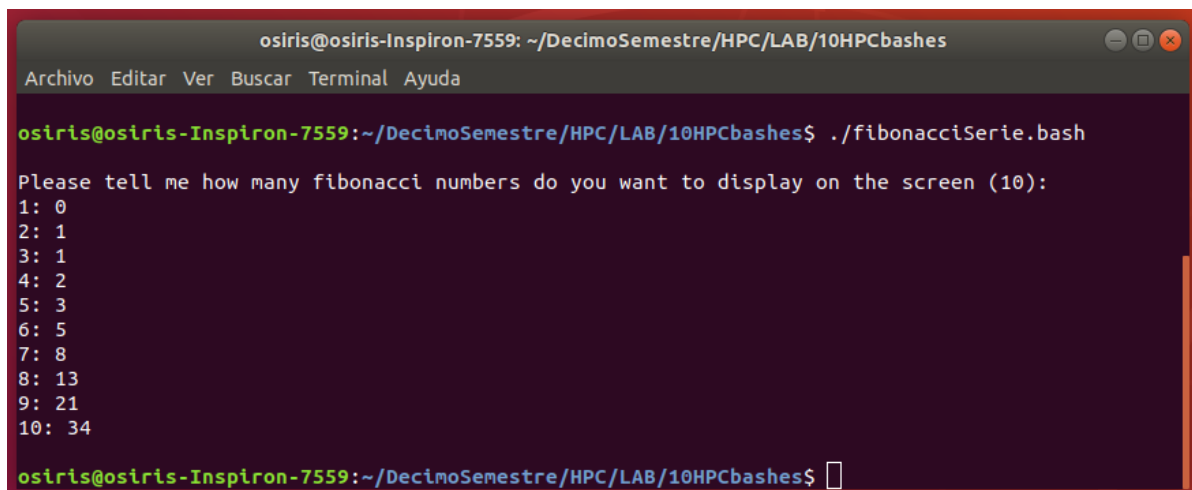
Listing 4: bash version: Example 4

```

1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 ## PRINT THE FIRST 10 ELEMENTS OF THE FIBONACCI SERIE

```

```
5 echo
6 read -p "Please tell me how many fibonacci numbers do you want to display on the screen (10):"
7
8 final=${final:-10}
9 first=0
10 second=1
11 auxiliar=0
12 count=1
13
14 if [ $final -ge 94 ] #-ge means greater or equal (inclusive)
15 then
16     ((final2=final))
17     final=93
18 fi
19
20 echo "1: " $first
21 echo "2: " $second
22
23 while [ $count -lt $((final-1)) ]; #-lt means less than (exclusive)
24 do
25     ((auxiliar= first))
26     ((first= second))
27     ((second= auxiliar + first))
28     echo "${count+2}: " $second
29     ((count++))
30 done
31
32 if [ $final -ge 93 ] #-ge means greater or equal (inclusive)
33 then
34     echo "Sorry, ${final2} fibonacci numbers can't be computed"
35 fi
36 echo
```



```
osiris@osiris-Inspiron-7559: ~/DecimoSemestre/HPC/LAB/10HPCbashes
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$ ./fibonacciSerie.bash

Please tell me how many fibonacci numbers do you want to display on the screen (10):
1: 0
2: 1
3: 1
4: 2
5: 3
6: 5
7: 8
8: 13
9: 21
10: 34

osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$
```

Executing in Terminal: OUTPUT

5. The next script creates a specific size file (**in Megabytes**) with random alphanumeric characters. If the user doesn't provide a specific size, the default size of 1Mb is taken.

Listing 5: bash version: Example 5

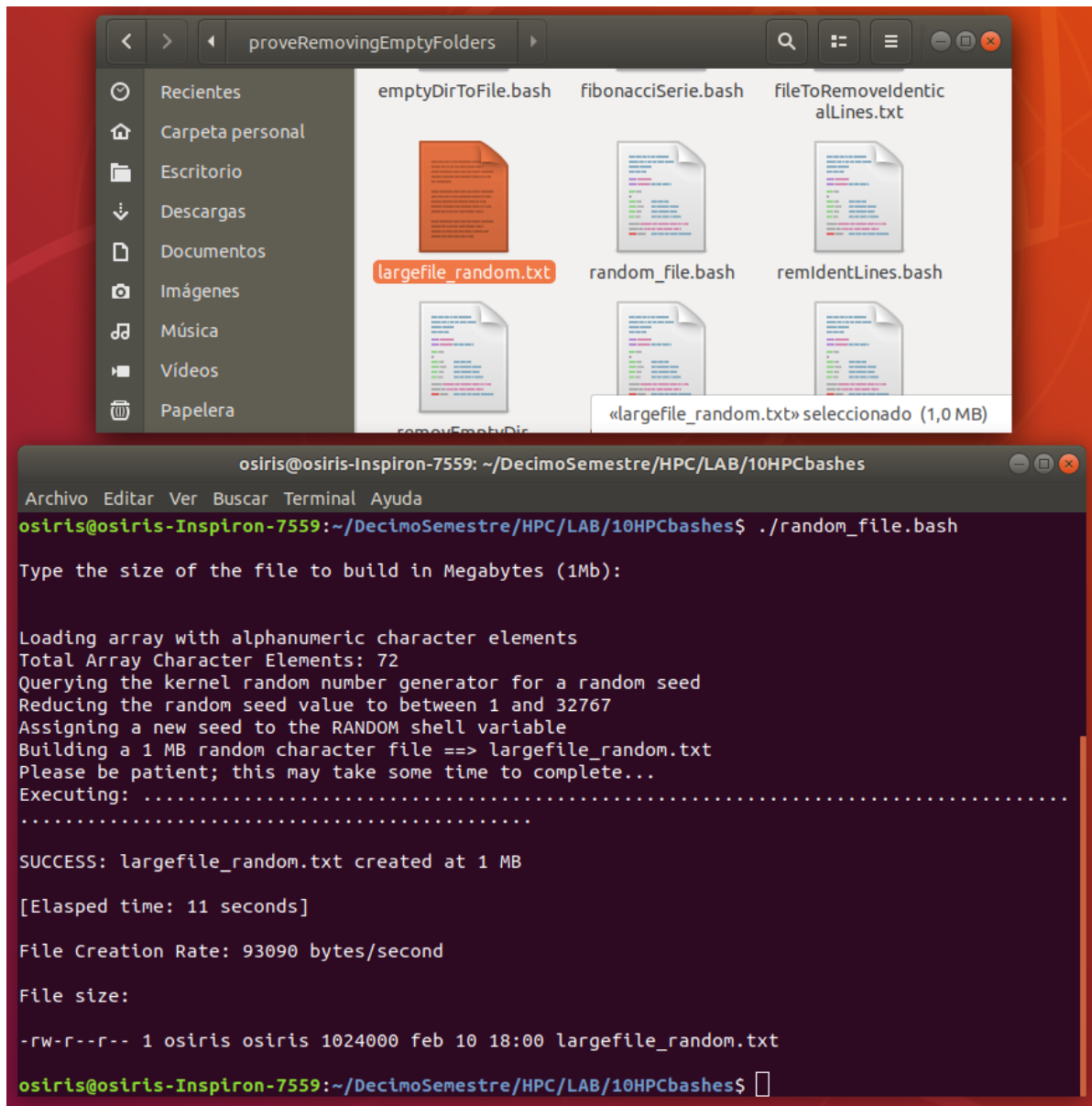
```
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 #
5 # SCRIPT: random_file.Bash
6 # AUTHOR: Randy Michael
7 # DATE: 8/3/2007
8 # REV: 1.0
9 # PLATFORM: Not Platform Dependent
10 #
11 # PURPOSE: This script is used to create a
12 #specific size file of random characters.
13 #The methods used in this script include
14 #loading an array with alphanumeric
15 #characters , then using the /dev/random
16 #character special file to seed the RANDOM
17 #shell variable , which in turn is used to
18 #extract a random set of characters from the
19 #KEYS array to build the OUTFILE of a
20 #specified MB size .
21 #
22 # set -x # Uncomment to debug this script
23 #
24 # set -n # Uncomment to check script syntax
25 #
26 # without any execution . Do not forget
27 #
28 # to put the comment back in or the
29 #
30 # script will never execute .
31 #
32 #####
33 # DEFINE FILES AND VARIABLES HERE
34 #####
35 #typeset -i MB_SIZE=$1
36 echo
37 read -p "Type the size of the file to build in Megabytes (1Mb): " MB_SIZE
38 echo
39 MB_SIZE=${MB_SIZE:-1}
40 typeset -i RN
41 typeset -i i=1
```

```
42 typeset -i X=0
43 OUTFILE=largefile_random.txt
44 >$OUTFILE
45 THIS_SCRIPT=$(basename $0)
46 CHAR_FILE=${WORKDIR}/char_file.txt
47 #####
48 # DEFINE FUNCTIONS HERE
49 #####
50 build_random_line ()
51 {
52 # This function extracts random characters
53 # from the KEYS array by using the RANDOM
54 # shell variable
55 C=1
56 LINE=
57 until (( C > 79 ))
58 do
59 LINE="${LINE}${KEYS[$(($RANDOM %X + 1))]}"
60 (( C = C + 1 ))
61 done
62 # Return the line of random characters
63 echo "$LINE"
64 }
65 #####
66 elapsed_time ()
67 {
68 SEC=$1
69 (( SEC < 60 )) && echo -e "[Elapsed time: \
70 $SEC seconds]\c"
71 (( SEC >= 60 && SEC < 3600 )) && echo -e \
72 "[Elapsed time: $(( SEC / 60 )) min $(( SEC % 60 )) sec]\c"
73 (( SEC > 3600 )) && echo -e "[Elapsed time: \
74 $(( SEC / 3600 )) hr $(( (SEC % 3600) / 60 )) min \
75 $(( (SEC % 3600) % 60 )) sec]\c"
76 }
77 #####
78 load_default_keyboard ()
79 {
80 # Loop through each character in the following list and
81 # append each character to the $CHAR_FILE file. This
82 # produces a file with one character on each line.
83 for CHAR in 1 2 3 4 5 6 7 8 9 0 q w e r t y u i o \
84 p a s d f g h j k l z x c v b n m \
85 Q W E R T Y U I O P A S D F G H J K L \
86 Z X C V B N M 0 1 2 3 4 5 6 7 8 9
87 do
```

```
88 echo "$CHAR" >> $CHAR_FILE
89 done
90 }
91 #####
92 usage ()
93 {
94 echo -e "\nUSAGE: $THIS_SCRIPT Mb_size"
95 echo -e "Where Mb_size is the size of the file to build\n"
96 }
97 #####
98 # BEGINNING OF MAIN
99 #####
100 #if (( $# != 1 ))
101 #then
102 #usage
103 #exit 1
104 #fi
105 # Test for an integer value
106 case $MB_SIZE in
107 [0-9]) : # Do nothing
108 ;;
109 *) usage
110 ;;
111 esac
112 # Test for the $CHAR_FILE
113 if [ ! -s "$CHAR_FILE" ]
114 then
115 echo -e "\nNOTE: $CHAR_FILE does not exist"
116 echo "Loading default keyboard data."
117 echo -e "Creating $CHAR_FILE...\c"
118 load_default_keyboard
119 echo "Done"
120 fi
121 # Load Character Array
122 echo -e "\nLoading array with alphanumeric character elements"
123 while read ARRAY_ELEMENT
124 do
125 (( X = X + 1 ))
126 KEYS[$X]=$ARRAY_ELEMENT
127 done < $CHAR_FILE
128 echo "Total Array Character Elements: $X"
129 # Use /dev/random to seed the shell variable RANDOM
130 echo "Querying the kernel random number generator for a random seed"
131 RN=$(dd if=/dev/random count=1 2>/dev/null \
132 | od -t u4 | awk '{print $2}' | head -n 1)
133 # The shell variable RANDOM is limited to 32767
```



```
134 echo "Reducing the random seed value to between 1 and 32767"
135 RN=$(( RN % 32767 + 1 ))
136 # Initialize RANDOM with a new seed
137 echo "Assigning a new seed to the RANDOM shell variable"
138 RANDOM=$RN
139 echo "Building a $MB_SIZE MB random character file ==> $OUTFILE"
140 echo "Please be patient; this may take some time to complete..."
141 echo -e ".\c"
142 # Reset the shell SECONDS variable to zero seconds.
143 SECONDS=0
144 TOT_LINES=$(( MB_SIZE * 12800 ))
145 until (( i > TOT_LINES ))
146 do
147   build_random_line >> $OUTFILE
148   (( $( ( i % 100 ) ) == 0 )) && echo -e ".\c"
149   (( i = i + 1 ))
150 done
151 # Capture the total seconds
152 TOT_SEC=$SECONDS
153 echo -e "\n\nSUCCESS: $OUTFILE created at $MB_SIZE MB\n"
154 elapsed_time $TOT_SEC
155 # Calculate the bytes/second file creation rate
156 (( MB_SEC = ( MB_SIZE * 1024000 ) / TOT_SEC ))
157 echo -e "\n\nFile Creation Rate: $MB_SEC bytes/second\n"
158 echo -e "File size:\n"
159 ls -l $OUTFILE
160 echo
161 #####
162 # END OF random_file.Bash SCRIPT
163 #####
```



Executing in Terminal: OUTPUT

- The next script take the name of a file as an argument and remove all identical lines inside the file maintaining the original order of the lines inside the file.

Listing 6: bash version: Example 6

```

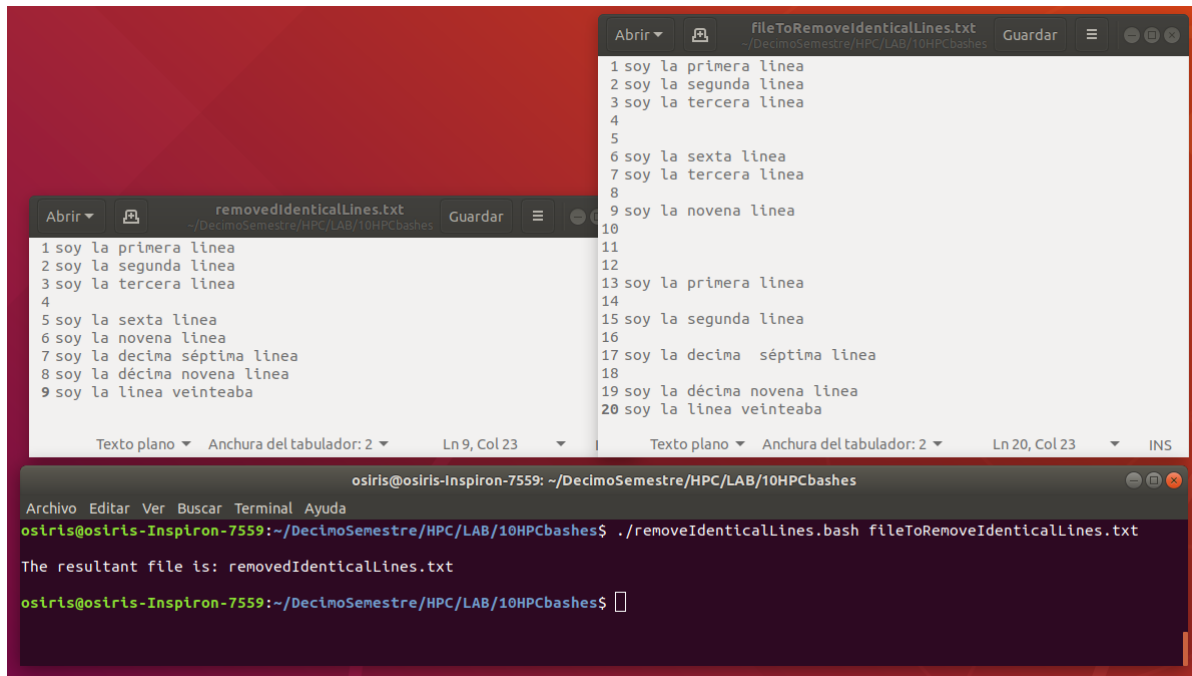
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 echo
5 file_out=removedIdenticalLines.txt
6
7 ARRAY=();

```

```

8 rm -f ${file_out}
9 while IFS=' ' read -r line || [[ -n "$line" ]]; do
10     if [[ ! "${ARRAY[@]}" =~ "$line" ]]; then
11         ARRAY+=("${ARRAY[@]}" "$line")
12         echo $line >> ${file_out}
13     fi
14 done < "$1"
15 echo "The resultant file is: ${file_out}"
16 echo

```



Executing in Terminal: OUTPUT

- The next script takes a path or a directory name and remove all empty sub directories.

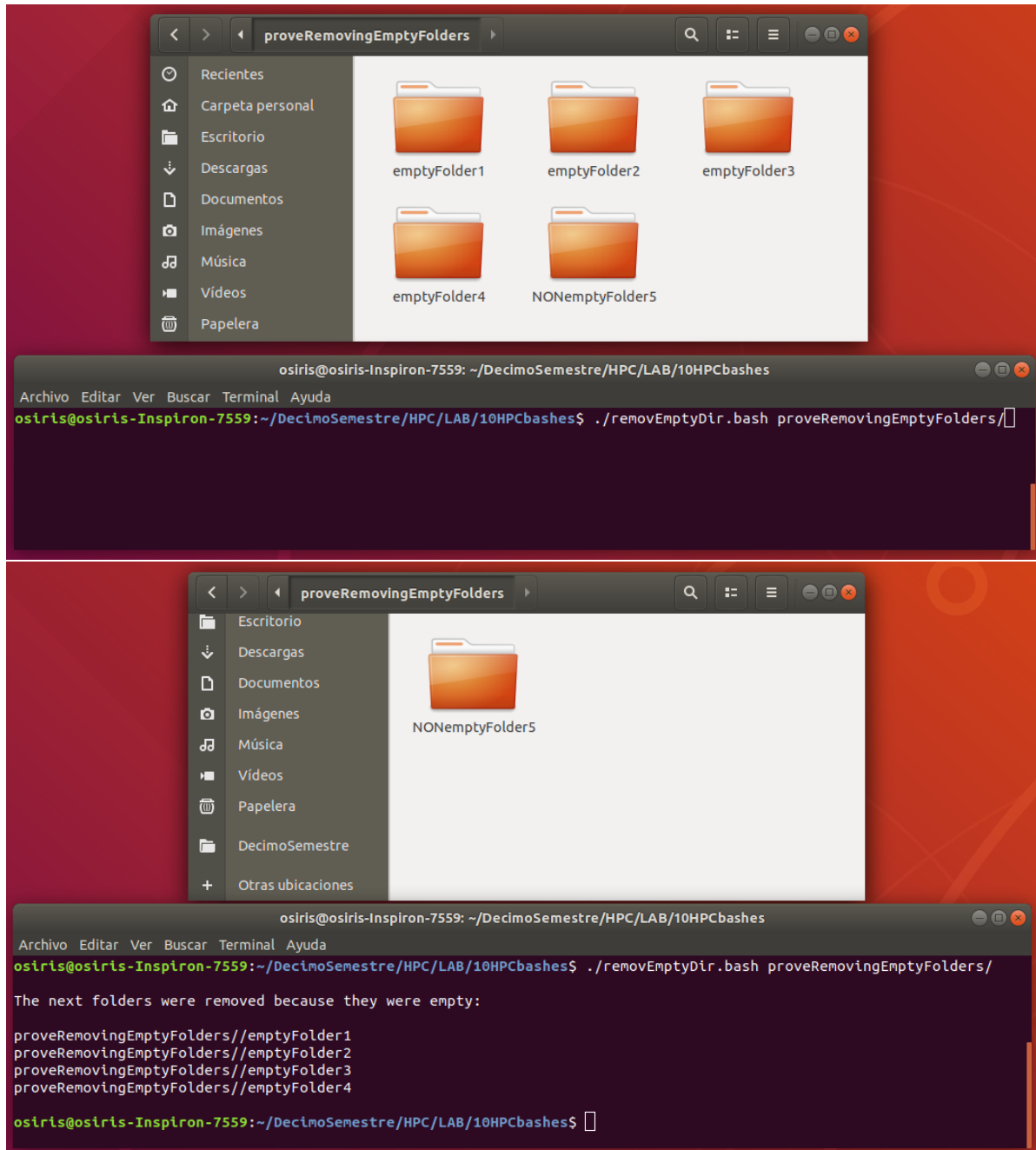
Listing 7: bash version: Example 7

```

1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4
5 LIST=$(ls -d $1/*)
6
7 echo
8 echo "The next folders were removed because they were empty: "
9 echo
10 for rdir in $LIST
11 do
12     if [ -n "$(find "$rdir" -maxdepth 0 -type d -empty 2>/dev/null)" ]; then

```

```
13     echo ${rdir}
14     rmdir ${rdir}
15 fi
16 done
17 echo
```

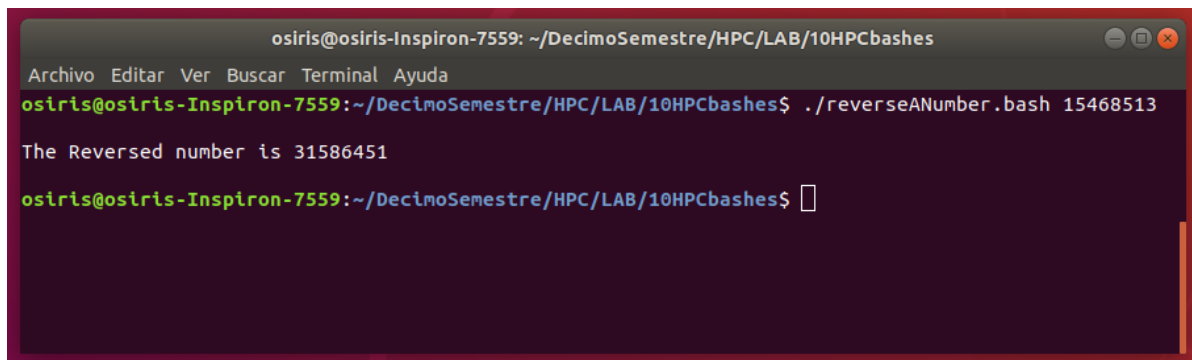


Executing in Terminal: OUTPUT

8. The next script takes a number as an input argument and return the reverse number as a result.

Listing 8: bash version: Example 8

```
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 echo
5 if [ $# -ne 1 ]
6 then
7     echo "Please provide the correct input in the below format."
8     echo "Usage: $0 number"
9     echo "    This script will reverse the given number."
10    echo "    For eg. $0 1234, will print 4321"
11    echo
12    exit 1
13 fi
14
15 n=$1
16 rev=0
17 sd=0
18
19 while [ $n -gt 0 ]
20 do
21     sd='expr $n % 10'
22     rev='expr $rev \* 10 + $sd'
23     n='expr $n / 10'
24 done
25 echo "The Reversed number is $rev"
26 echo
```

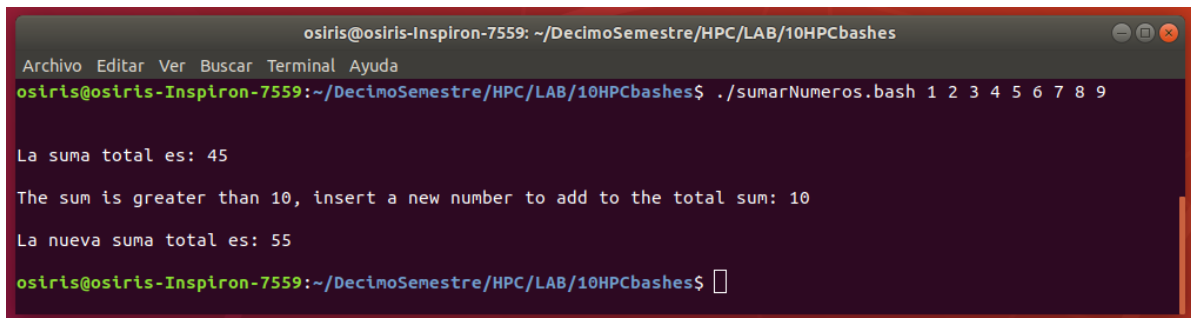
A screenshot of a terminal window with a dark background and light-colored text. The window title is "osiris@osiris-Inspiron-7559: ~/DecimoSemestre/HPC/LAB/10HPCbashes". The terminal shows the command prompt "osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes\$./reverseANumber.bash 15468513". The output of the script is "The Reversed number is 31586451". The prompt is followed by a cursor. The terminal window has standard Linux window controls (minimize, maximize, close) in the top right corner.

Executing in Terminal: OUTPUT

9. The next script takes an unspecified number of command line arguments (**up to 9**) of ints and finds their sum. The code add a number to the sum only if the number is greater than 10.

Listing 9: bash version: Example 9

```
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 echo
5
6 sumaTotal=0
7
8 if [ $# -gt 9 ]; then
9     echo "Sorry, you can only pass up to 9 parameters"
10    echo
11 else
12     echo
13     for i in "$@"
14     do
15         sumaTotal=$((sumaTotal+i))
16     done
17     echo "La suma total es: ${sumaTotal}"
18     echo
19     if [ $sumaTotal -gt 10 ]; then
20         read -p "The sum is greater than 10, insert a new number to add to the"
21         sumaTotal=$((sumaTotal+lastNumber))
22         echo
23         echo "La nueva suma total es: ${sumaTotal}"
24         echo
25     fi
26 fi
```



```
osiris@osiris-Inspiron-7559: ~/DecimoSemestre/HPC/LAB/10HPCbashes
Archivo Editar Ver Buscar Terminal Ayuda
osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$ ./sumarNumeros.bash 1 2 3 4 5 6 7 8 9

La suma total es: 45

The sum is greater than 10, insert a new number to add to the total sum: 10

La nueva suma total es: 55

osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$
```

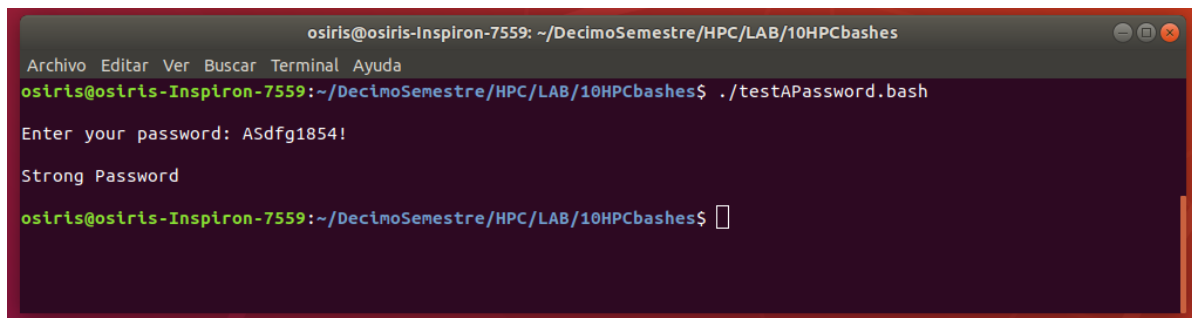
Executing in Terminal: OUTPUT

10. The next script takes a password and validate its strength based on:

- Length: minimum of 8 characters.
- Contain both alphabet and number.
- Include both the small and capital case letters.

Listing 10: bash version: Example 10

```
1 #!/bin/bash
2 #Author: Osiris Rom n
3 #Math and SC school
4 #Password Validation Script
5 echo
6 read -p "Enter your password: " password
7 echo
8 len="${#password}"
9
10 if test $len -ge 8 ; then
11     echo "$password" | grep -q [0-9]
12     if test $? -eq 0 ; then
13         echo "$password" | grep -q [A-Z]
14         if test $? -eq 0 ; then
15             echo "$password" | grep -q [a-z]
16             if test $? -eq 0 ; then
17                 echo "Strong Password"
18             else
19                 echo "Weak Password -> Should include a lower case letter"
20             fi
21         else
22             echo "Weak Password -> Should include a capital case letter."
23         fi
24     else
25         echo "Weak Password -> Should use numbers in your password."
26     fi
27 else
28     echo "Weak Password -> Password length should have at least 8 characters."
29 fi
30 echo
```



```
osiris@osiris-Inspiron-7559: ~/DecimoSemestre/HPC/LAB/10HPCbashes
Archivo Editar Ver Buscar Terminal Ayuda
osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$ ./testAPassword.bash
Enter your password: Asdfg1854!
Strong Password
osiris@osiris-Inspiron-7559:~/DecimoSemestre/HPC/LAB/10HPCbashes$
```

Executing in Terminal: OUTPUT

Signature of the student