Module Title:	Operating Systems & Computer Networks
Module Code:	CST2555
Module Leader:	Dr Ian Mitchell

Bash Script
Coursework 1

Student Number: Anonymous
Tutor:

Computer Science

2021-22

Contents

Chapter 1

Create Files

The listing in fig. 1.1 shows the code submitted to complete the task of creating files.
Here is a list of assumptions about this script:

- User is content with Directories are cleaned once the user enters a new set inputs hence directory CWFiles has been directed to be cleaners after each successful input.
- It is assumed that the user may input invalid parameters therefore there are checks in place to prevent from crashing.
- User has previous knowledge on how to run bash script files in English.
- The user understands error messages.

```
#!/bin/bash
##creates three levels of directories
##textfiles name is dependant on the second level directory followed by third i.e. numbers inputted by the  user
##numbers inside textfile must be same as the name of textfile
##clean direcotry to overried the existing files inside the directory
##produces erros messages based on parameters entered,inputs lower than 0,greater than 10,and if m>n
m=$1
n=$2


dir="CWFiles"
text=".txt"

##create three level directories
createFiles(){
        mkdir $dir
        for (( x = $1; x <= $2; x++))
        do
                mkdir $dir/$x
                for (( y = $1; y <= $2; y++ ))
                do
                        mkdir $dir/$x/$y
                        for (( z = $1; z <= $2; z++ ))
                        do
                                ##inside text file  the file name is printed
                                exec 5>&1
                                exec 1>$dir/$x/$y/$x$y$z$text
                                echo $x$y$z
                                exec 1<&5
                        done
                done
        done
        echo "Files" $m  "to" $n "have been created!"
}

##clear directory
if [ -d "CWFiles" ]; then
        rm -r "CWFiles"
fi

##checking for valid inputs before function createFiles is executed
invalid="You must input two numbers!"
bigValue="Inputs cannot be greater than 10 "
input="First input cannot be greater than second input"
smallValue="Inputs cannot be lower than 0"

if [ $# -ne 2 ]; then
         echo $invalid

    elif [ "$m" -lt 0 ] || [ "$n" -lt 0 ] ; then
             echo $smallValue

    elif [ "$m" -gt "$n" ]; then
             echo $input

    elif [ "$m" -gt 10 ] || [ "$n" -gt 10 ]; then
             echo $bigValue

    else createFiles $m $n
fi
```

Figure 1.1: This is the listing for creating files.

Chapter 2

Read Files

The listing in fig. 2.1 shows the code submitted to complete the task of reading files.
Here is a list of assumption about this script:

- It is assumed users input numbers in ./createFiles.sh *before* they enter the same inputs in readFiles.sh; only then will the sum of the number of files will be produced.
- The sum of files in readFiles.sh would only be what the users' latest input in createFiles.sh any previous input in createFiles.sh will be not be accounted of as it overwritten after each successful input in createFiles.sh
- User will understand error messages prompted by the script

```bash
#!/bin/bash
##gets highest and lowest value from an array which gives the user inputs
##if inital input of createFiles matches with readFiles matches continue with readFiles function
##get textfile content i.e. first line of each files contaning file number and sums total



##prints creatFiles input  without the CWFiles
dir=$(ls -d CWFiles/* | awk -F "/"  '{print $2}' )

##in an array these function scans for inside thearray for the lowest ie the first input of user and the higest element ie users second input
getMin() {

         min=("$@")
         for d in "${min[@]}"
         do
                if [ $d -lt "$min" ];then
                        min=$d
                fi
        done
        echo $min
}
##gets value of a vairable from inside a function and assigns it to a a new variable
min=$(getMin $dir)

getMax() {

        max=("$@")
        for d in "${max[@]}"
        do
                if [ $d -gt "$max" ]; then
                        max=$d
                fi
        done
        echo $max
}
##gets value of a vairable from inside a function and assigns it to a a new variable
max=$(getMax $dir)



##scans for path gets only first line of text file and sums first line of each of the files  that has been created and produces sum
readFiles() {
        dir="CWFiles"
        text=".txt"
        sum=0
        for (( x = $1; x <= $2; x++))
        do

             for (( y = $1; y <= $2; y++ ))
                do

                        for (( z = $1; z <= $2; z++ ))
                        do

                                        numbers=$dir/$x/$y/$x$y$z$text

                                        exec 5<&0
                                        exec 0<$numbers
                                        content=""

                                        while read line
                                        do
                                                content=$(echo $line)
                                        done
                                        exec 0>&5
                                        numtxt=$(sed -n '1p' $numbers)
                                        num=$(echo $numtxt | sed 's/^0*//')

                                        sum=$[ $sum + $(($num)) ]

                        done
                done
        done
        echo "Total file numbers:" $sum
}


m=$1
n=$2
##produces errors messgaes
input="First input cannot be greater than second input"
empty="You must input two numbers!"
matcherror="Ensure you have inputted in createfiles correctly before inputting in readfiles"
bigValue="Inputs cannot be greater than 10 please ensure you have inputted the same values as you did in ./createFiles.sh"
smallValue="Inputs cannot be lower than 0 please ensure you have inputted the same values as you did in ./createFiles.sh"
if [ $# -ne 2 ]; then
        echo $empty
elif [ "$m" -lt 0 ] || [ "$n" -lt 0 ] ; then
        echo $smallValue
elif [ "$m" -gt "$n" ]; then
        echo $input
elif [ "$m" -gt 10 ] || [ "$n" -gt 10 ]; then
        echo $bigValue
##check if the values enetered at readFiles.sh matches the ones in createFiles.sh if succesfull continues with readFiles

elif [ $min == "$m" ] && [ $max == "$n" ]; then
        readFiles $m $n
else
        echo $matcherror
fi
```

Figure 2.1: This is the listing for creating files.

Chapter 3

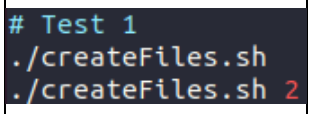
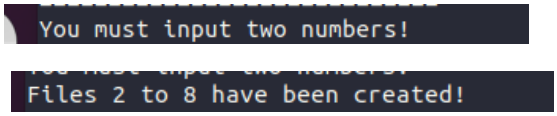
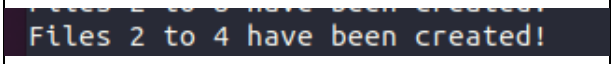
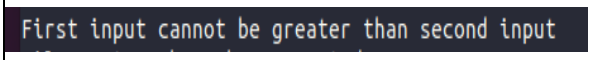
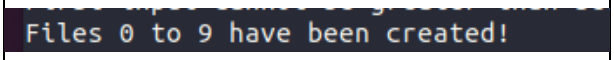
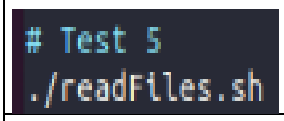
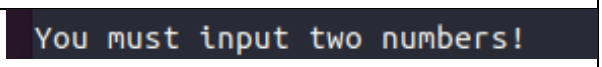
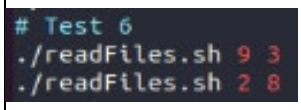
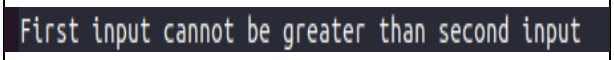
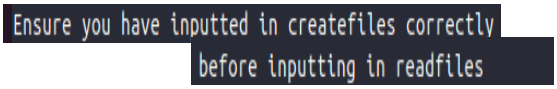
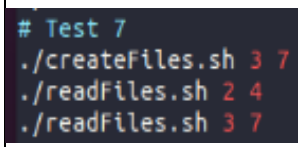
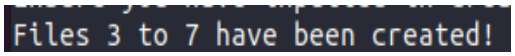
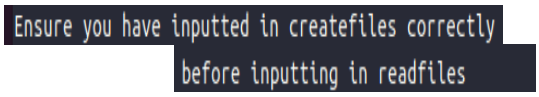
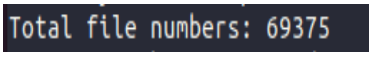
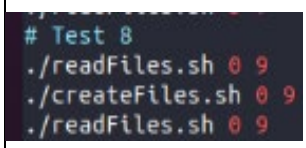
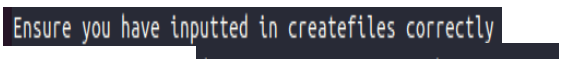
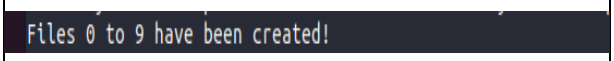
Evaluation and Testing

This section should run the code in listing 3.1. The output is explained and each of the 8 tests and their outcomes. Below a table is include with the output of each of the tests in separate listings.

```
silvia@silvia-VirtualBox:~/CST2555/Bash CW2$ tar -cf cw2555.tar readFiles.sh createFiles.sh
silvia@silvia-VirtualBox:~/CST2555/Bash CW2$ ./test.sh
============================
|  Start Create File Test   |
============================
You must input two numbers!
Files 2 to 8 have been created!
Files 2 to 4 have been created!
First input cannot be greater than second input
Files 0 to 9 have been created!
============================
| Finished Create File Test |
============================

============================
|    Start Read File Test   |
============================
You must input two numbers!
First input cannot be greater than second input
Ensure you have inputted in createfiles correctly before inputting in readfiles
Files 3 to 7 have been created!
Ensure you have inputted in createfiles correctly before inputting in readfiles
Total file numbers: 69375
Ensure you have inputted in createfiles correctly before inputting in readfiles
Files 0 to 9 have been created!
Total file numbers: 499500
============================
|  Finished Read File Test  |
============================
silvia@silvia-VirtualBox:~/CST2555/Bash CW2$
```

Figure 3.1: Test file ./test.sh

Test	Expected	Outcome
	• Two parameters are required Files will not be created. • Files will be created and prompt user with a success message.	
	• Directory will be cleaned removing all previous files created in CWFiles new files will be created and prompt user with a success message.	
	• First parameter must be lower than second parameter user will be prompted with an error message.	
	• Directory will be cleaned removing all previous files created in CWFiles from the test ./createFiles.sh 2 4, new files will be created and prompt user with a success message.	
	• Two parameters are required in order to get the sum from createFiles.sh user will be prompted with an error message.	
	• The first parameter cannot be greater than the second parameter (as createFiles.sh also does not allow this) hence use will be prompted with an error message • User will be prompted with an error message notifying the parameters entered in readFiles.sh is not the same as createFiles.sh	 
	• Directory will be cleaned removing all previous files created in CWFiles new files will be created and prompt user with a success message. • User will be prompted with an error message notifying the parameters entered in readFiles.sh is not the same as createFiles.sh • As parameters entered in createFiles.sh and readFiles.sh matches the sum of all files created will be displayed	  
	• User will be prompted with an error message notifying the parameters entered in readFiles.sh is not the same as createFiles.sh • Directory will be cleaned removing all previous files created in CWFiles, and new files will be created and prompt user with a success message. • As parameters entered in createFiles.sh and readFiles.sh matches the sum of all files created will be displayed.	  

Chapter 4

Conclusions

To succesfully execute the files users must understand the purpose of a tar ball (tar -cf cw2555.tar readFile.sh createFiles.sh) and the command in test.sh which untars the files (tar -xf cw555.tar) and run the script which is used to test the files createFiles.sh and readFiles.sh . To summarisethe createFiles.sh the script for the user it's purpose is for the user to generate files withing the parameter range the user entered and delete previously created files, in essence overwrite, user must enter the numbers in the m < n format. As for the readFiles.sh its script read the parameters entered in createFiles.sh and if th inputs match the parameters enetered in readFiles.sh then it succesfully creates the sum of all files created from the parameter range.

It is assummed that the user is satisfied to overwrite files everytime a new parameter is entered in createFiles.sh, in addition to being aware that the inputs entered in readFiles.sh must be the same as the the previously enteres parameters in createFiles.sh.The user must also ensure that they enter two pramaeters in createFiles.sh before readFiles.sh to avoid errors.

Bibliography

[1] R. Blum and C. Bresnahan, *Linux command line and shell scripting bible*. Indianapolis, In: John Wiley & Sons, Inc, 2015.

[2] J. L. Coupanec, “LeCoupa/awesome-cheatsheets,” *GitHub*, Jan. 07, 2022. https://github.com/LeCoupa/awesome-cheatsheets/blob/master/languages/bash.sh (accessed Jan. 07, 2022).

[3] C. Ramey, B. Fox, and B. Gough, *GNU bash reference manual*. Bristol: Network Theory Ltd, 2009.