

ESE2025_ Bash scripts

Name: Shreya Mamadapur
Instructor: Takis Zourntos

Student ID: C0774035

Introduction

This report introduces an interpreter or command language in Linux - Bash. Bash stands for Bourne Again Shell. It was written by Brian Fox as a replacement for Bourne Shell. In this report, we will go through four bash scripts.

Discussion

Bash is the default command language for Linux. We can check our default interpreter in Linux by typing the command `echo $SHELL` in our terminal. There are many advantages of writing bash commands in a script file. We do not have to run commands repetitively in our terminal, we can simply execute the written bash script and save a lot of time and manual entries.

We write our script in a simple nano editor. What makes our normal nano editor file a bash file is the "shebang". Shebang is the first thing that needs to be written in the script file. Shebang = `#!/bin/bash`. Or `#!/bin/sh` if we want to use shell interpreter.

To execute the script we can simply use `./scriptname.sh`. But to do this the file must be an executable file. So, `chmod` can be used to set permissions accordingly before executing. We can also use `bash <filename>` to run the script. In this case, the file need not be made executable file. Note that these scripts are just like normal files in Linux. It is always good to give `.sh` extension just for the ease for users to identify the script files.

By solving the following four examples we will get to know more commands like `sort`, `wc -l` and `ack(print)` etc. We will also know the syntax of using `if..else`, `for`. Please read through the explanation and the code in the images. The result is in the Terminal screenshot images.

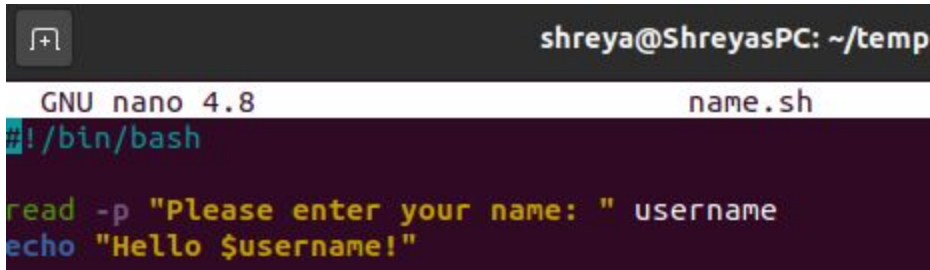
a) Bash script that accepts the user's name and prints out a "Hello <user name>!" greeting

In this script file, I'm taking input from the user for her/his name using `read -p` command and displaying a greeting using `echo` command.

`$username` is to say print the value in the variable `username`

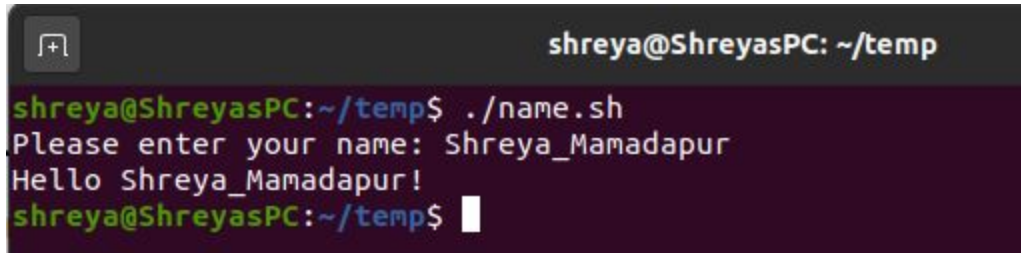
ESE2025_ Bash scripts

Bash script:



```
shreya@ShreyasPC: ~/temp
GNU nano 4.8                                name.sh
#!/bin/bash
read -p "Please enter your name: " username
echo "Hello $username!"
```

Terminal:



```
shreya@ShreyasPC: ~/temp
shreya@ShreyasPC:~/temp$ ./name.sh
Please enter your name: Shreya_Mamadapur
Hello Shreya_Mamadapur!
shreya@ShreyasPC:~/temp$
```

b) write a Bash script that searches the user's Documents/ folder for a file name "ese2025.txt"; if the file does not exist, create the file with the string "Found First!". If the file exists, append the string "Found Again!" to the file. End your script by sending the contents of the ~/Document/ese2025.txt file to standard output.

In this script, I'm initializing a variable dir with my desired path. Variable file contains the actual desired file.

I'm using a conditional statement to check if my file not exists.

touch command is to create a file.

echo "whatever you wanna say" > \$filename.txt #to type in the file

Note that > replaces the file contents, while >> is used to append/add some desired contents

ESE2025_ Bash scripts

Bash script:

```
shreya@ShreyasPC: ~/temp
GNU nano 4.8 filesearch.sh
#!/bin/bash

dir=~/Documents/folder
file=$dir/ese2025.txt

if [[ ! -e $file ]] #if file does not exists
then
    touch $file
    echo "Found first!" > $file
else
    echo "Found again!!" >> $file
fi
```

Terminal:

```
shreya@ShreyasPC: ~/temp
shreya@ShreyasPC:~/temp$ cat ~/Documents/folder/ese2025.txt
cat: /home/shreya/Documents/folder/ese2025.txt: No such file or directory
shreya@ShreyasPC:~/temp$ ./filesearch.sh
shreya@ShreyasPC:~/temp$ cat ~/Documents/folder/ese2025.txt
Found first!
shreya@ShreyasPC:~/temp$ ./filesearch.sh
shreya@ShreyasPC:~/temp$ cat ~/Documents/folder/ese2025.txt
Found first!
Found again!!
shreya@ShreyasPC:~/temp$
```

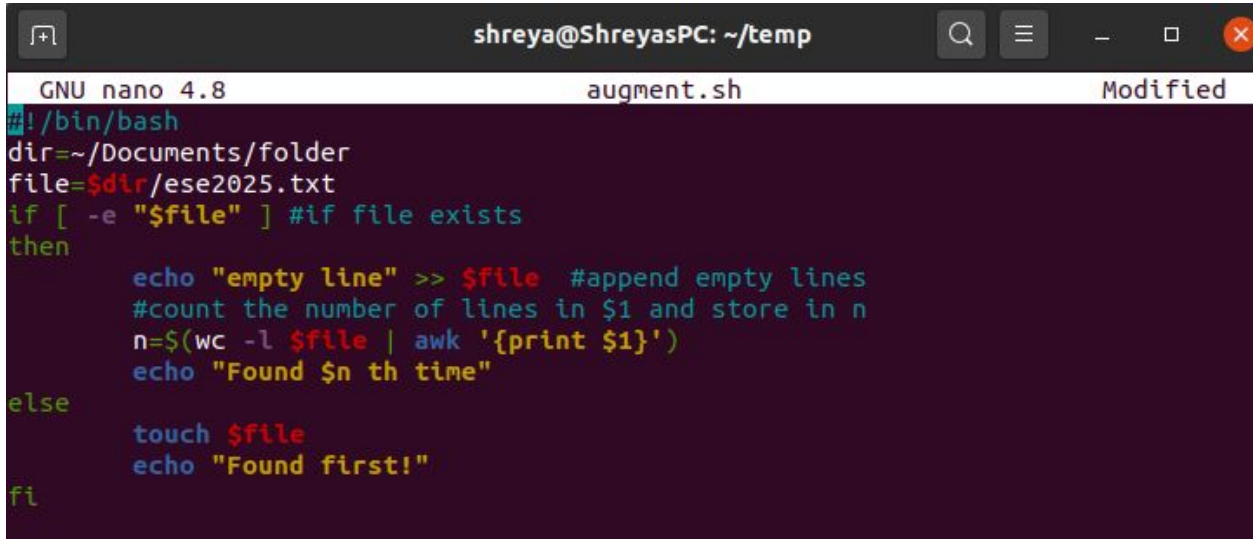
c) augment the functionality of the Bash script from b) by changing "Found Again!" to "Found for the <n>th time!" where n represents the number of times that the file has been found.

The if condition checks if \$file exists. If it exists we need to count n (number of time file is being found). To do that, I'm first adding empty line in the \$file. Then using wc(word count) command I'm counting the number of lines and printing the n value. N value = the number of empty lines. (i.e each time the script is executed the number of empty lines adds up)

awk is a function which has some C like function definitions for eg: print

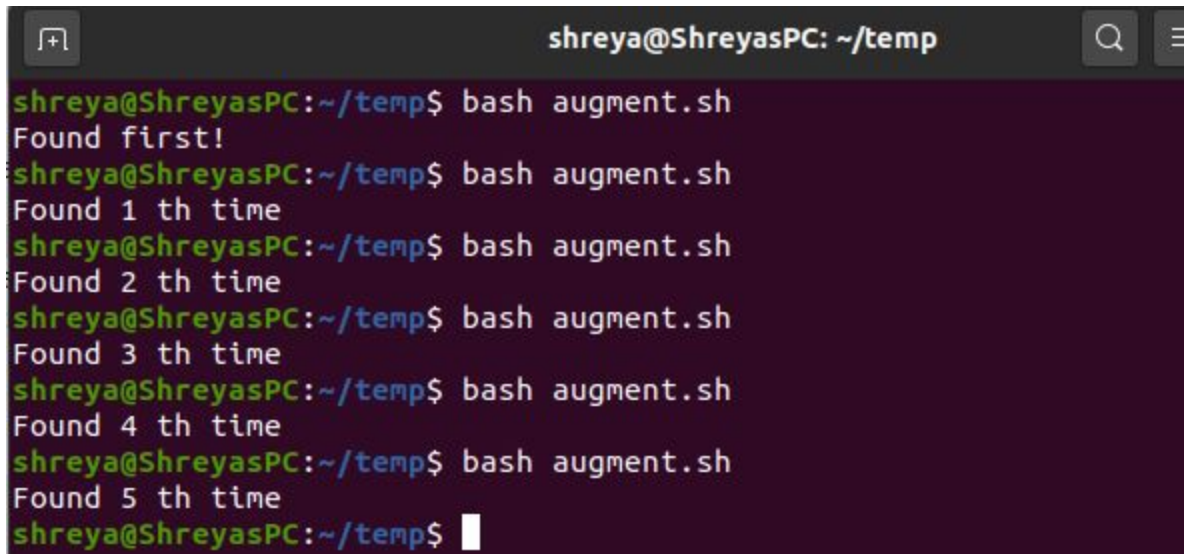
ESE2025_ Bash scripts

Bash script:



```
shreya@ShreyasPC: ~/temp
GNU nano 4.8                                augment.sh                                Modified
#!/bin/bash
dir=~/Documents/folder
file=$dir/e2025.txt
if [ -e "$file" ] #if file exists
then
    echo "empty line" >> $file #append empty lines
    #count the number of lines in $1 and store in n
    n=$(wc -l $file | awk '{print $1}')
    echo "Found $n th time"
else
    touch $file
    echo "Found first!"
fi
```

Terminal:



```
shreya@ShreyasPC:~/temp$ bash augment.sh
Found first!
shreya@ShreyasPC:~/temp$ bash augment.sh
Found 1 th time
shreya@ShreyasPC:~/temp$ bash augment.sh
Found 2 th time
shreya@ShreyasPC:~/temp$ bash augment.sh
Found 3 th time
shreya@ShreyasPC:~/temp$ bash augment.sh
Found 4 th time
shreya@ShreyasPC:~/temp$ bash augment.sh
Found 5 th time
shreya@ShreyasPC:~/temp$
```

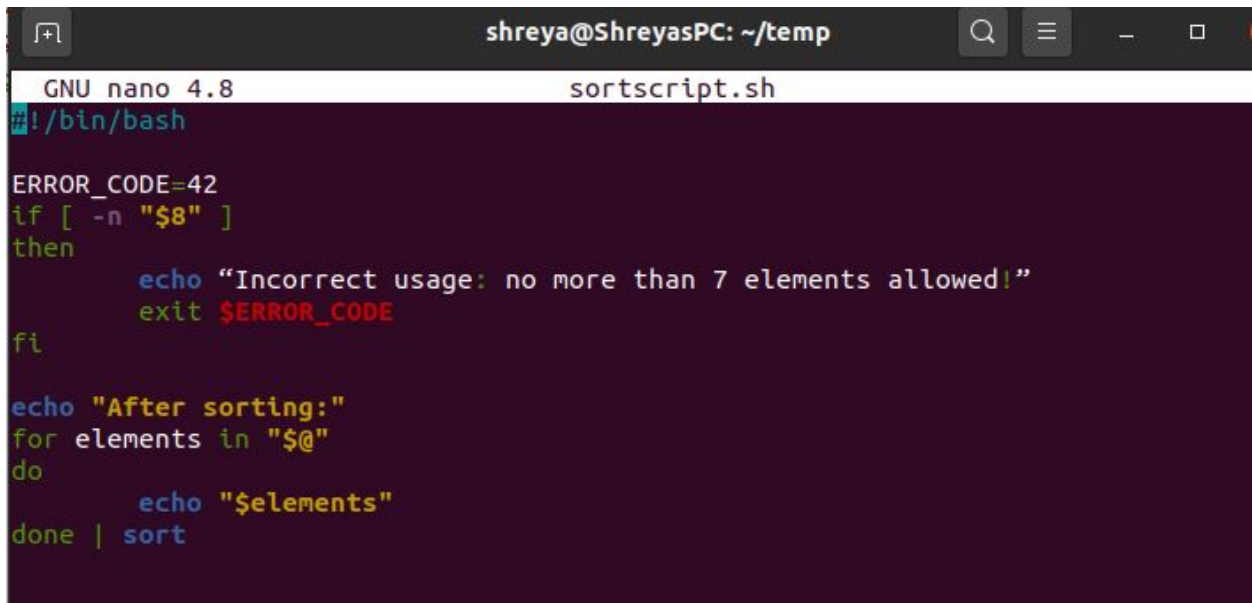
d) Bash script that sorts names entered by the user as command-line arguments and gives error_code 42 if more than 7 arguments are given.

If there's an 8th argument, echo an error message and exit with an error code.

Otherwise, use for loop to get all the elements and pipeline it to sort command.

ESE2025_ Bash scripts

Bash script:

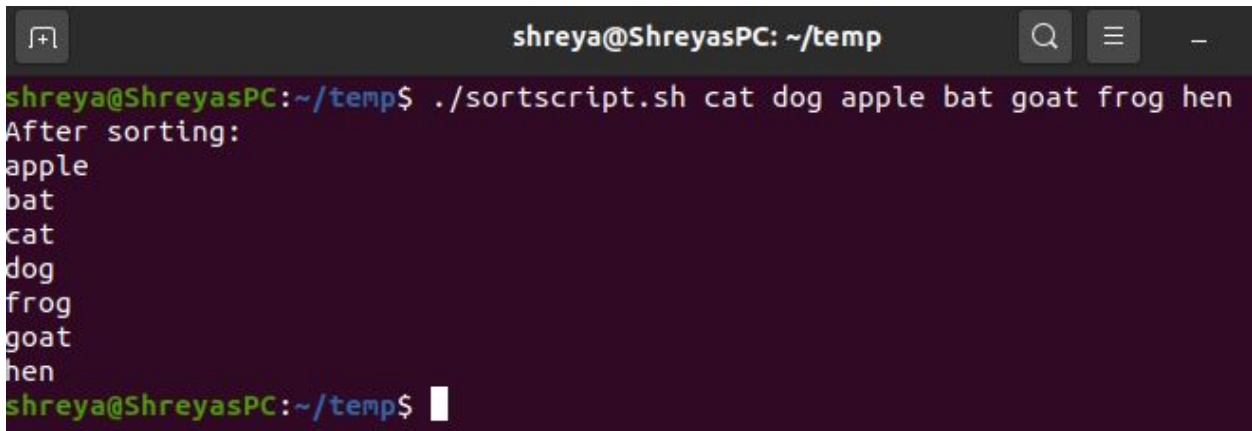


```
shreya@ShreyasPC: ~/temp
GNU nano 4.8 sortscript.sh
#!/bin/bash

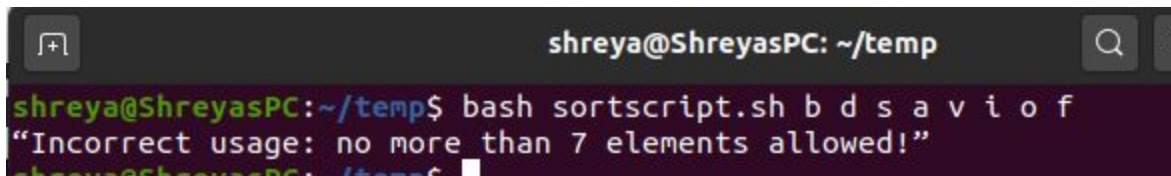
ERROR_CODE=42
if [ -n "$8" ]
then
    echo "Incorrect usage: no more than 7 elements allowed!"
    exit $ERROR_CODE
fi

echo "After sorting:"
for elements in "$@"
do
    echo "$elements"
done | sort
```

Terminal:



```
shreya@ShreyasPC: ~/temp
shreya@ShreyasPC:~/temp$ ./sortscript.sh cat dog apple bat goat frog hen
After sorting:
apple
bat
cat
dog
frog
goat
hen
shreya@ShreyasPC:~/temp$
```



```
shreya@ShreyasPC: ~/temp
shreya@ShreyasPC:~/temp$ bash sortscript.sh b d s a v i o f
"Incorrect usage: no more than 7 elements allowed!"
shreya@ShreyasPC:~/temp$
```

Conclusion

ESE2025_ Bash scripts

This report helps beginners kickstart with a new command language called Bash. Bash is a simple yet handy tool in Linux for novice programmers as well.

Appendix

#Bashscript1

```
#!/bin/bash
```

```
read -p "Please enter your name: " username
echo "Hello $username!"
```

#Bashscript2

```
#!/bin/bash
```

```
dir=~/Documents/folder
```

```
file=$dir/ese2025.txt
```

```
if [[ ! -e $file ]] #if file does not exists
then
    touch $file
    echo "Found first!" > $file
else
    echo "Found again!!" >> $file
fi
```

#Bashscript3

```
#!/bin/bash
```

```
dir=~/Documents/folder
```

```
file=$dir/ese2025.txt
```

```
if [ -e "$file" ] #if file exists
then
    echo "empty line" >> $file #append empty lines
    #count the number of lines in $1 and store in n
    n=$(wc -l $file | awk '{print $1}')
    echo "Found $n th time"
else
    touch $file
```

ESE2025_ Bash scripts

```
        echo "Found first!"  
fi
```

```
#Bashscript4  
#!/bin/bash
```

```
Error_code=42  
if [ -n "$8" ]  
then  
    echo "Incorrect usage: no more than 7 elements allowed!"  
    Exit $Error_code  
fi
```

```
echo "After sorting:"
```

```
for elements in "$@"  
do  
    echo "$elements"  
done | sort
```