



Operating System Project

(CSD-222)

Write a Shell Script Using Array to Store All Lines Count.

Submitted By

Ritwik Duggal (185528)

Himanshi (185539)

Abhishek kumar (185540)

Submitted To

Dr. Pardeep Singh

Department of Computer Science and Engineering

INTRODUCTION

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

This project is on Bash script.

BASH SCRIPT

What are they exactly?

A Bash script is a plain text file which contains a series of commands. These commands are a mixture of commands we would normally type ourselves on the command line (such as `ls` or `cp` for example) and commands we could type on the command line but generally wouldn't (you'll discover these over the next few pages). An important point to remember though is:

Anything you can run normally on the command line can be put into a script and it will do exactly the same thing. Similarly, anything you can put into a script can also be run normally on the command line and it will do exactly the same thing.

You don't need to change anything. Just type the commands as you would normally and they will behave as they would normally. It's just that instead of typing them at the command line we are now entering them into a plain text file. In this sense, if you know how to do stuff at the command line

then you already know a fair bit in terms of Bash scripting.

It is convention to give files that are Bash scripts an extension of `.sh` (`myscript.sh` for example). As you would be aware (and if you're not maybe you should consider reviewing our [Linux Tutorial](#)), Linux is an extensionless system so a script doesn't necessarily have to have this characteristic in order to work.

How do they work?

In the realm of Linux (and computers in general) we have the concept of programs and processes. A program is a blob of binary data consisting of a series of instructions for the CPU and possibly other resources (images, sound files and such) organised into a package and typically stored on your hard disk. When we say we are running a program we are not really running the program but a copy of it which is called a process. What we do is copy those instructions and resources from the hard disk into working memory (or RAM). We also allocate a bit of space in RAM for the process to store variables (to hold temporary working data) and a few flags to allow the operating system (OS) to manage and track the process during its execution.

Essentially a process is a running instance of a program.

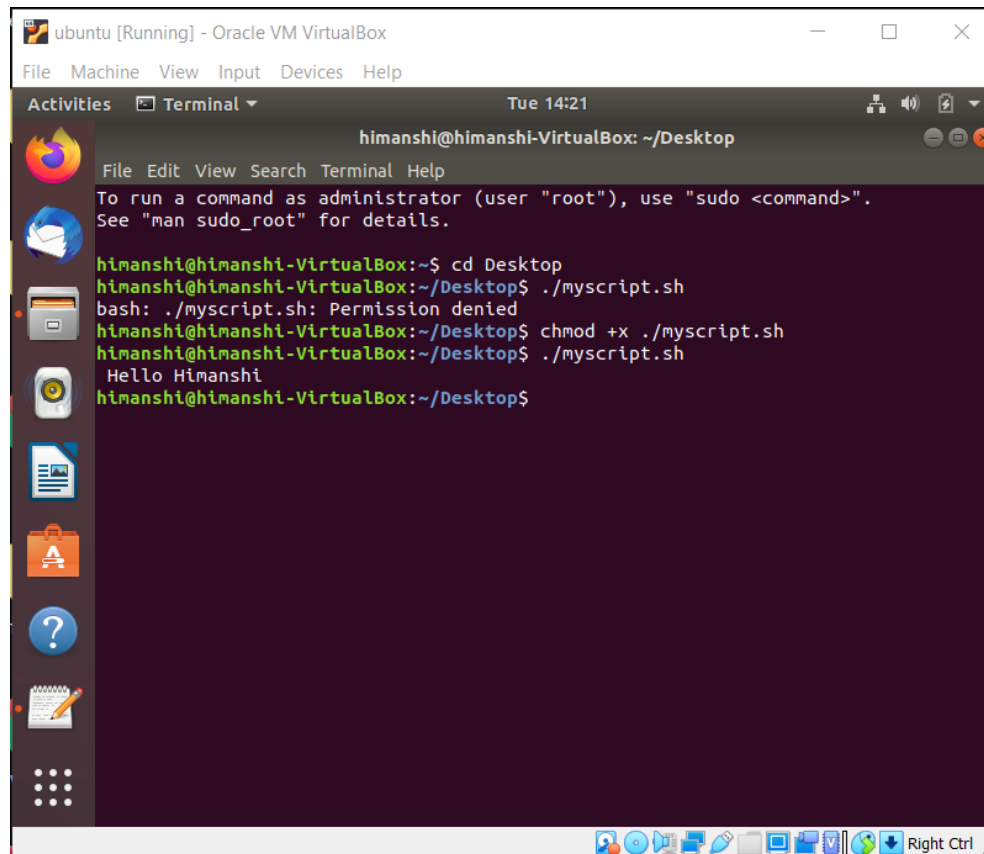
There could be several processes representing the same program running in memory at the same time. For example I could have two terminals open and be running the command `cp` in both of them. In this case there would be two `cp` processes currently existing on the system. Once they are finished running the system then destroys them and there are no longer any processes representing the program `cp`.

When we are at the terminal we have a Bash process running in order to give us the Bash shell. If we start a script running it doesn't actually run in that process but instead starts a new process to run inside. We'll demonstrate this in the next section on variables and its implications should become clearer. For the most part you don't need to worry too much about this phenomenon however.

How do we run them?

Running a Bash script is fairly easy. Another term you may come across is executing the script (which means the same thing). Before we can execute a script it must have the execute permission set (for safety reasons

this permission is generally not set by default). If you forget to grant this permission before running the script you'll just get an error message telling you as such and no harm will be done.



```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Tue 14:21
himanshi@himanshi-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

himanshi@himanshi-VirtualBox:~$ cd Desktop
himanshi@himanshi-VirtualBox:~/Desktop$ ./myscript.sh
bash: ./myscript.sh: Permission denied
himanshi@himanshi-VirtualBox:~/Desktop$ chmod +x ./myscript.sh
himanshi@himanshi-VirtualBox:~/Desktop$ ./myscript.sh
Hello Himanshi
himanshi@himanshi-VirtualBox:~/Desktop$
```

Figure 1: Terminal

SOFTWARE USED

1. Virtual Box

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

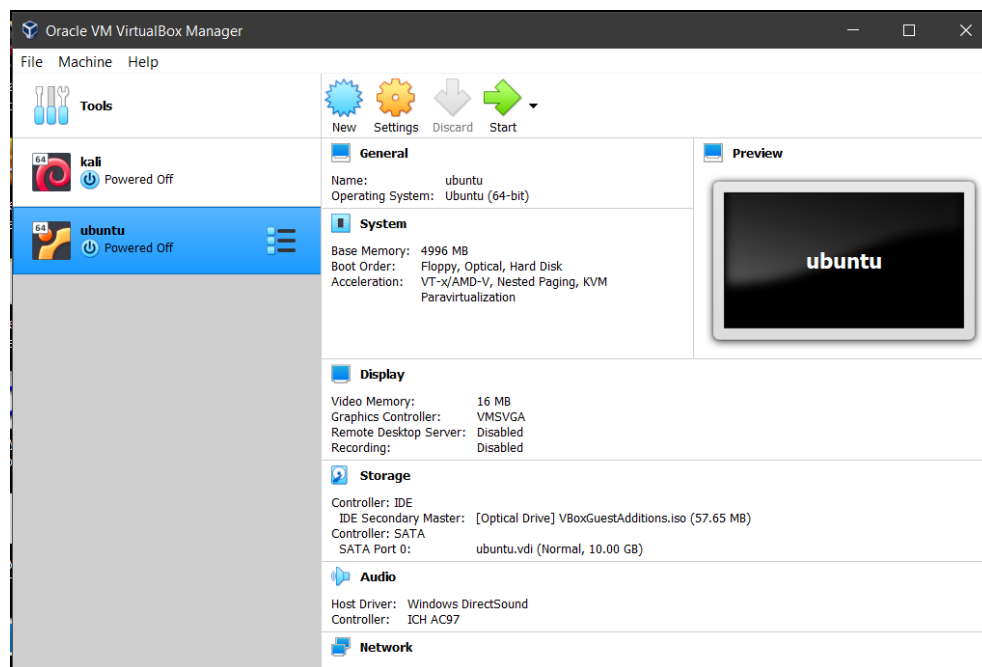


Figure 2: Virtual Box

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the

product always meets professional quality criteria.

2. Ubuntu OS

Ubuntu is a Linux-based operating system. It is designed for computers, smartphones, and network servers. The system is developed by a UK based company called Canonical Ltd. All the principles used to develop the Ubuntu software are based on the principles of Open Source software development.

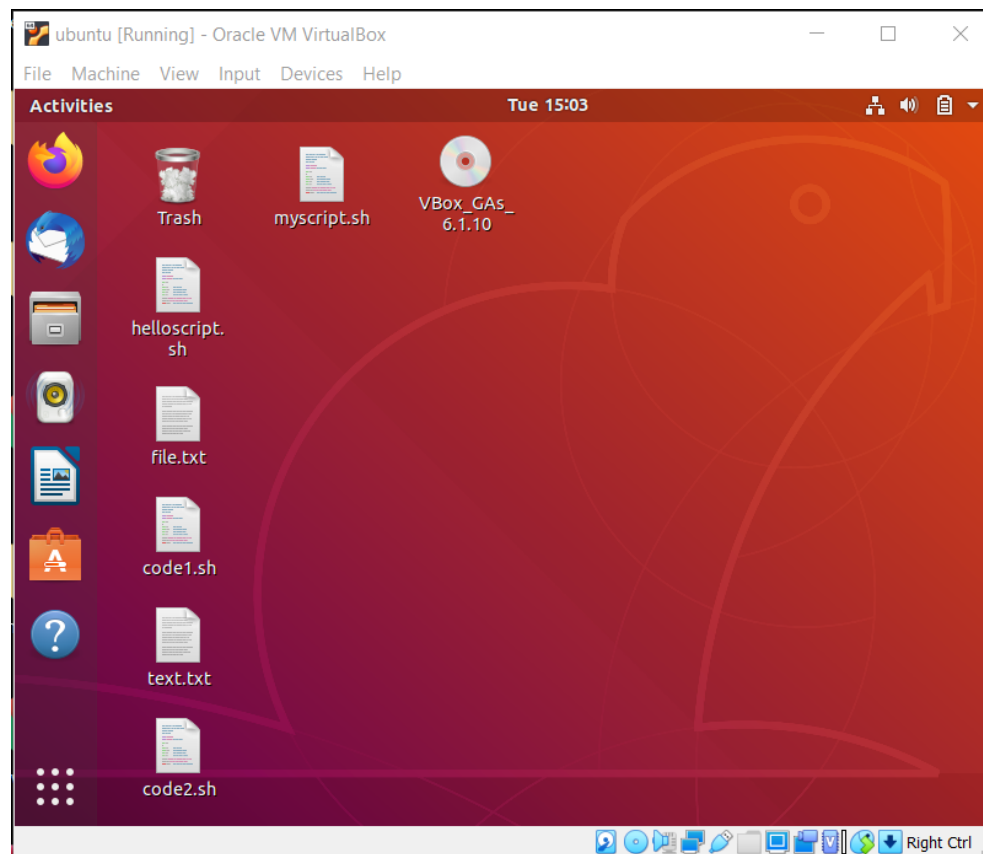


Figure 3: Ubuntu OS

COMMANDS USED

1. echo

echo command in linux is used to display line of text/string that are passed as an argument . This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.

Syntax :

echo [option] [string]

2. read

read command in Linux system is used to read from a file descriptor. Basically, this command read up the total number of bytes from the specified file descriptor into the buffer. If the number or count is zero then this command may detect the errors. But on success, it returns the number of bytes read. Zero indicates the end of the file. If some errors found then it returns -1.

Syntax:

read

3. cat

The cat (short for “concatenate”) command is one of the most frequently used command in Linux/Unix like operating systems. cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

4. seq

seq command in Linux is used to generate numbers from FIRST to LAST in steps of INCREMENT. It is a very useful command where we had to generate list of numbers in while, for, until loop.

Syntax:

seq [OPTION]... LAST

or

```
seq [OPTION]... FIRST LAST
or
seq [OPTION]... FIRST INCREMENT LAST
```

5. wc

wc stands for word count. As the name implies, it is mainly used for counting purpose. It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments. -l: This option prints the number of lines present in a file. With this option wc command displays two-columnar output, 1st column shows number of lines present in a file and 2nd itself represent the file name.

With one file name
wc -l state.txt

6. for loop

The for loop operate on lists of items. It repeats a set of commands for every item in a list. Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN. Syntax

```
for var in word1 word2 ...wordn
do
Statement to be executed
done
```

7. while loop

The bash while loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition.

syntax for "while loop"

```
while [ condition ]
do
command1
```



```
command2
command3
done
```

CODES FOR STORING ALL LINES COUNT USING ARRAY

Code 1

```
#!/bin/bash

store_all_lines_count(){
    f=$1
    index=0
    echo "File content of " $f "is : "
    while read line
    do
        Line[$index]="$line"
        index=$((index+1))
    done < $f

    for i in $(seq 0 $(( ${#Line[@]} - 1 )))
    do
        echo "i=$i - ${Line[$i]}"
    done
    echo -e "Total lines \c"
    wc -l $f
    echo -e "\n"
    echo -e "Array length is : " ${#Line[@]}
    echo -e "\n"
}

i=0
echo "Enter the file name: "
read file
```

```
store_all_lines_count $file
```

Algorithm:

Code1.sh

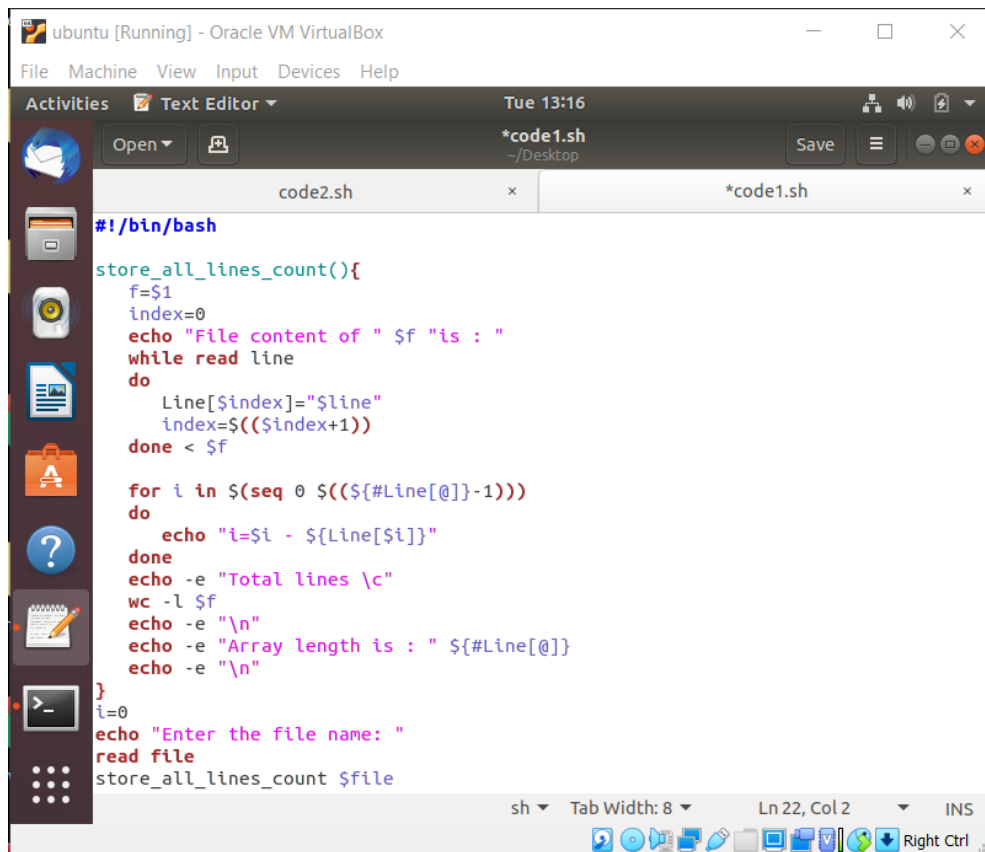
```
Store_all_lines_count function()
```

1. Declare a variable f which stores the calling file.
2. Print file contents of the given file.
3. Run while loop until read line .
4. Stores every single line of a file in an array variable named as Line.
5. Run for loop and print each line stored in array.
6. Print the \total line count" using wc command with -l flag.
7. Print the array length and compare it.

```
main:
```

1. Enter file name and read it in <file>.
2. Call function store_all_lines_count for a given <file >.

Attach screenshots of code 1 and it's output



The screenshot shows a VirtualBox window titled 'ubuntu [Running] - Oracle VM VirtualBox'. Inside the window is an Ubuntu desktop environment. The top panel includes a menu bar with 'File', 'Machine', 'View', 'Input', 'Devices', and 'Help'. Below the menu bar is a top bar with 'Activities', 'Text Editor', and a clock showing 'Tue 13:16'. The 'Text Editor' window has two tabs: 'code2.sh' and '*code1.sh'. The '*code1.sh' tab is active and shows a shell script. The script defines a function 'store_all_lines_count()' that takes a file path as an argument, reads the file line by line into an array, and then iterates through the array to print each line and the total count. The script also prompts the user to enter a file name and calls the function with that name. The bottom status bar shows 'sh', 'Tab Width: 8', 'Ln 22, Col 2', and 'INS'.

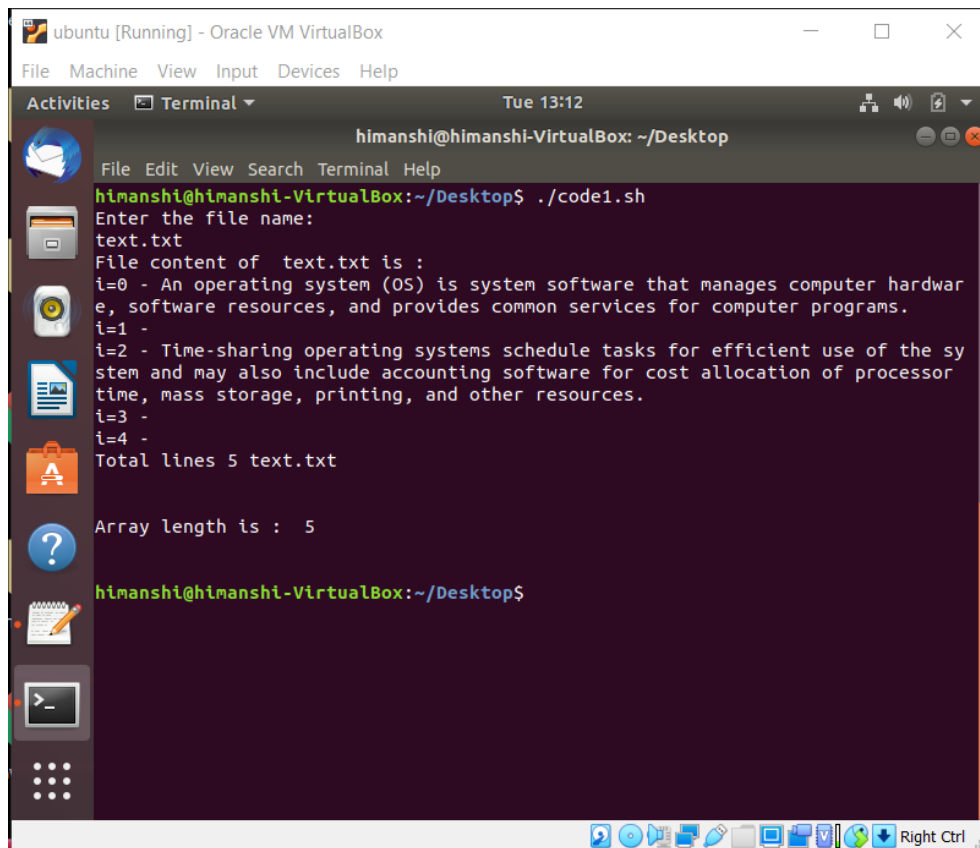
```
#!/bin/bash

store_all_lines_count(){
    f=$1
    index=0
    echo "File content of " $f "is : "
    while read line
    do
        Line[$index]=$line
        index=$((index+1))
    done < $f

    for i in $(seq 0 $(( ${#Line[@]} - 1 )) )
    do
        echo "i=$i - ${Line[$i]}"
    done
    echo -e "Total lines \c"
    wc -l $f
    echo -e "\n"
    echo -e "Array length is : " ${#Line[@]}
    echo -e "\n"
}

i=0
echo "Enter the file name: "
read file
store_all_lines_count $file
```

Figure 4: Code 1



The image shows a terminal window titled "ubuntu [Running] - Oracle VM VirtualBox". The window has a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". Below the menu bar is a toolbar with "Activities", "Terminal", and a clock showing "Tue 13:12". The terminal itself has a title bar "himanshi@himanshi-VirtualBox: ~/Desktop" and a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal output is as follows:

```
himanshi@himanshi-VirtualBox:~/Desktop$ ./code1.sh
Enter the file name:
text.txt
File content of text.txt is :
i=0 - An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.
i=1 -
i=2 - Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.
i=3 -
i=4 -
Total lines 5 text.txt

Array length is : 5

himanshi@himanshi-VirtualBox:~/Desktop$
```

Figure 5: Output of Code 1

Code 2

```
#!/bin/bash

echo "Enter the file name: "
read filename

ARRAY=( $(cat $filename | tr '\n' ' ') )
# testing
for i in $(seq 0 $(( ${#ARRAY[@]}-1 )))
do
    echo "i=$i - ${ARRAY[$i]}"
done

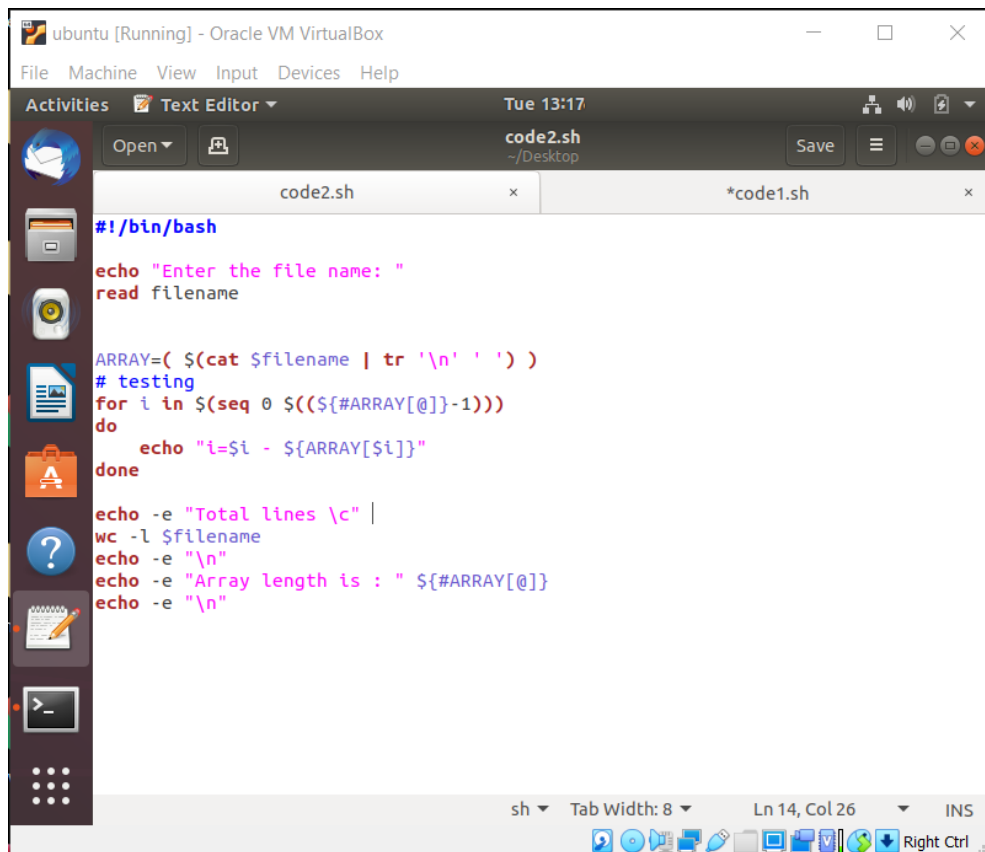
echo -e "Total lines \c"
wc -l $filename
echo -e "\n"
echo -e "Array length is : " ${#ARRAY[@]}
echo -e "\n"
```

Algorithm:

Code2.sh

1. Enter file name and read it in <filename>.
2. Store each line as an element in array <array>.
3. Run for loop and print each line stored in array.
4. Print the "total line count" using wc command with -l flag.
5. Print the array length and compare it.

Attach screenshots of code 2 and it's output



The screenshot shows a VirtualBox window titled "ubuntu [Running] - Oracle VM VirtualBox". Inside the window is an Ubuntu desktop environment. The top panel includes a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". Below the menu bar is a top bar with "Activities", "Text Editor", and the date "Tue 13:17". The "Text Editor" window has a title bar with "code2.sh" and a path "~/Desktop". It contains two tabs: "code2.sh" and "*code1.sh". The "code2.sh" tab is active and displays the following shell script:

```
#!/bin/bash

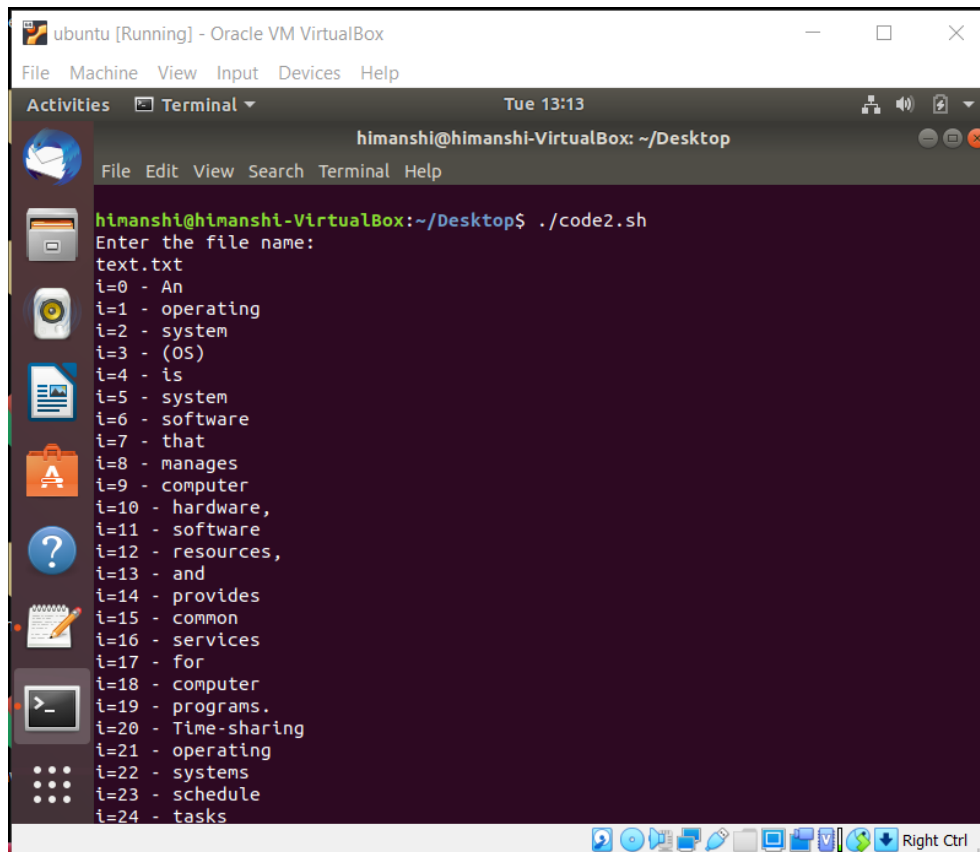
echo "Enter the file name: "
read filename

ARRAY=( $(cat $filename | tr '\n' ' ') )
# testing
for i in $(seq 0 ${#ARRAY[@]}-1))
do
    echo "i=$i - ${ARRAY[$i]}"
done

echo -e "Total lines \c" |
wc -l $filename
echo -e "\n"
echo -e "Array length is : " ${#ARRAY[@]}
echo -e "\n"
```

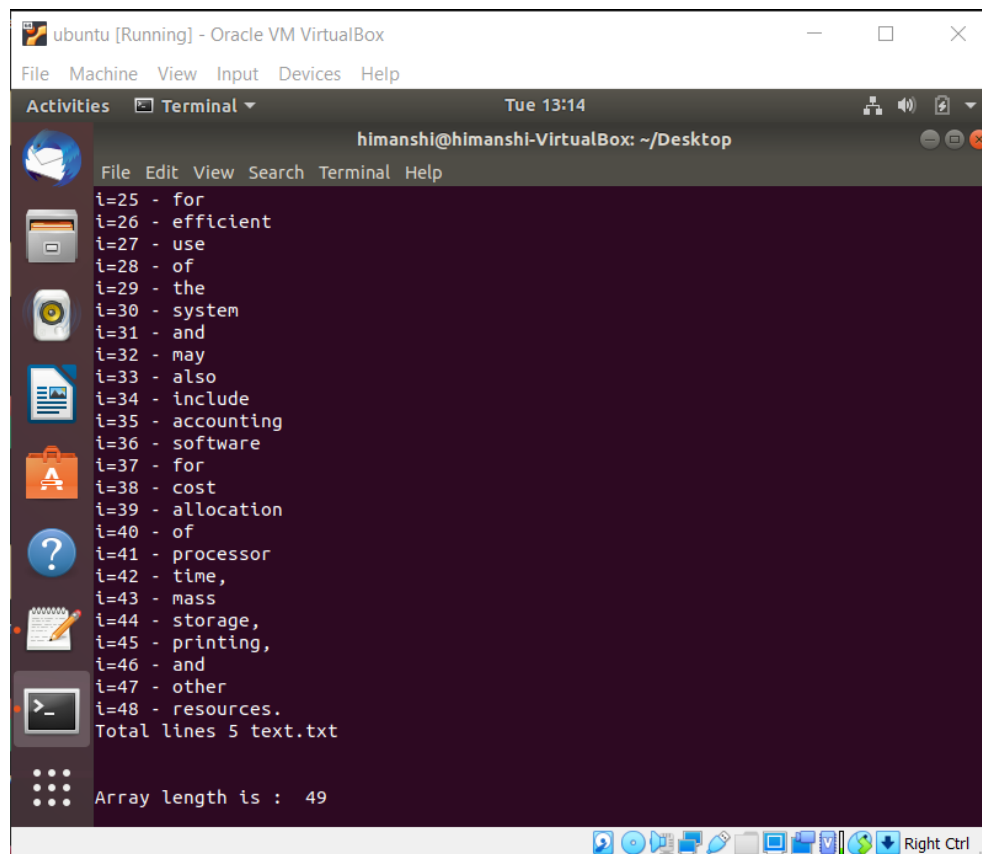
The bottom of the window shows a status bar with "sh", "Tab Width: 8", "Ln 14, Col 26", and "INS". A system tray at the bottom right contains various icons and the text "Right Ctrl".

Figure 6: Code 2



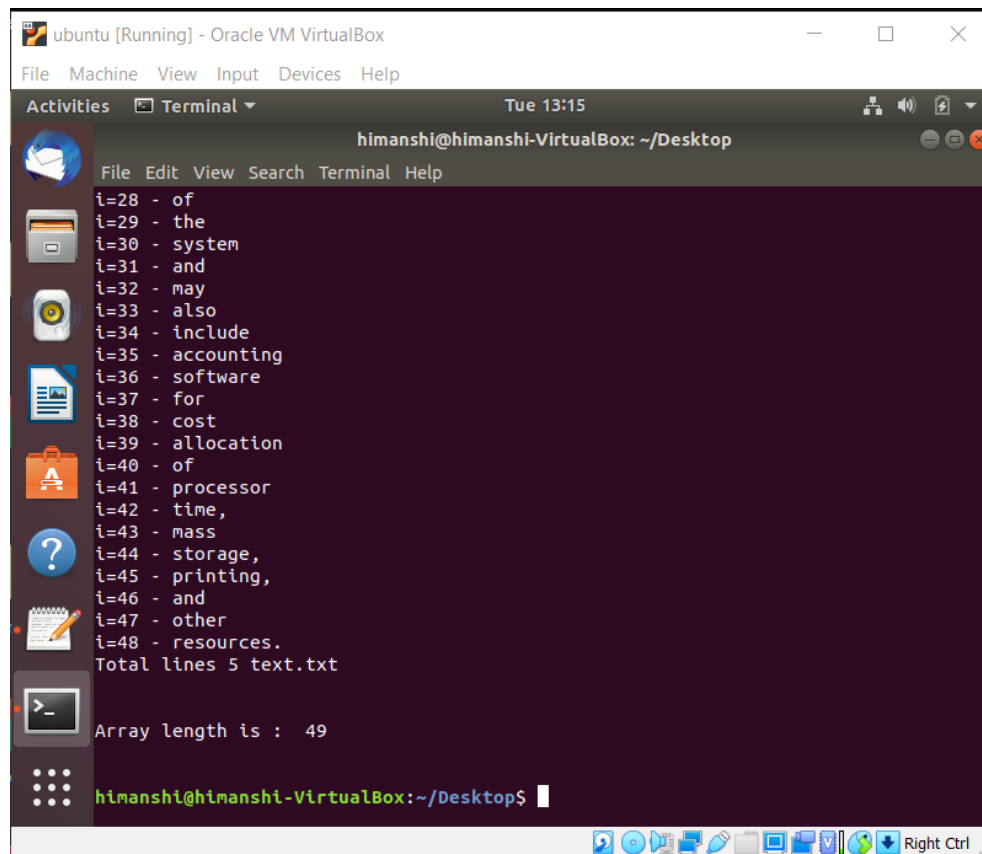
```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Tue 13:13
himanshi@himanshi-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
himanshi@himanshi-VirtualBox:~/Desktop$ ./code2.sh
Enter the file name:
text.txt
i=0 - An
i=1 - operating
i=2 - system
i=3 - (OS)
i=4 - is
i=5 - system
i=6 - software
i=7 - that
i=8 - manages
i=9 - computer
i=10 - hardware,
i=11 - software
i=12 - resources,
i=13 - and
i=14 - provides
i=15 - common
i=16 - services
i=17 - for
i=18 - computer
i=19 - programs.
i=20 - Time-sharing
i=21 - operating
i=22 - systems
i=23 - schedule
i=24 - tasks
```

Figure 7: Output of Code 2.a



```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Tue 13:14
himanshi@himanshi-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
i=25 - for
i=26 - efficient
i=27 - use
i=28 - of
i=29 - the
i=30 - system
i=31 - and
i=32 - may
i=33 - also
i=34 - include
i=35 - accounting
i=36 - software
i=37 - for
i=38 - cost
i=39 - allocation
i=40 - of
i=41 - processor
i=42 - time,
i=43 - mass
i=44 - storage,
i=45 - printing,
i=46 - and
i=47 - other
i=48 - resources.
Total lines 5 text.txt
Array length is : 49
```

Figure 8: Output of Code 2.b



```
ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Tue 13:15
himanshi@himanshi-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
i=28 - of
i=29 - the
i=30 - system
i=31 - and
i=32 - may
i=33 - also
i=34 - include
i=35 - accounting
i=36 - software
i=37 - for
i=38 - cost
i=39 - allocation
i=40 - of
i=41 - processor
i=42 - time,
i=43 - mass
i=44 - storage,
i=45 - printing,
i=46 - and
i=47 - other
i=48 - resources.
Total lines 5 text.txt
Array length is : 49
himanshi@himanshi-VirtualBox:~/Desktop$
```

Figure 9: Output of Code 2.c

COMPARISON

Code1

Time complexity: $O(n)$

Where n is number of lines in the file

Since it is using "wc -l" command which read each characters and count the number of in the file.

Memory: $O(1)$

It is only counting the number of newline character.

Constraint:

Sometimes in a file we use many newline character in the end of the data or file. It will count all of the extra blank space at the end as a line.

Code2

Time complexity: $O(m)$

Where m is number of characters in the file

Since it is using tr command to strip off newline character from the file and give each line as an element to the array. And finally we counting the length of the array.

Memory: $O(m)$

Where m is the number of line in the file.

Constraint:

It will only count the actual line in the file.i.e number of sentences since it strip off all the newline character from the file.