# Software Systems Lab: OutLab LaTeX

Name: Atishay Jain

Roll no: 210050026

September 6, 2022

# Contents

# 1 Introduction

This report consists of work done by me under **CS 251** course. This is a *Software Systems Lab* course that teaches many different kinds of programming utilities that are used by programmers quite often. This course is all about to help the students in the later courses.

## 1.1 Course Content

| S.N. | Topic |
|---|---|
| 1 | HTML/CSS and Java Script |
| 2 | Bash Basics + Git + Markdown |
| 3 | SED and AWK |
| 4 | LaTeX Basic + Advanced |
| 5 | Python Basics and Advanced |
| 6 | Web-scraping / DB / Networking |
| 7 | Java Basics |
| 8 | Make/Cmake |
| 9 | Doxygen and Sphinx |

Table 1: Topics taught under CS 251 course

Although there are many topics, but this report will be mainly based on my works and learning about **Linux**, **Bash**, **Sed** and **awk**.

## 1.2 Goal

- Presenting your work using HTML, CSS, Drawing Software, LaTeX, gnuplot
- Code Warrior via reading & writing code, profiling, Debugging, learning about Editors, IDE, Compilation, Version Control, Bash, Awk etc.
- Learning about Open Source Operating System - **Linux**
- Prepare students with the vocabulary of CS tools such that later concepts can be easily explored

# 2 Linux - An Open Source Operating System

**Linux**[1] was developed and publicly released by "Linus Torvalds" in September 1991. Linux is a free and open-source operating system, unlike either Windows OS or macOS. Various Linux Distros are available and are free to use. Typically the Linux OS is employed for cloud and server software. Some popular Linux distros are *Ubuntu, Debian, Fedora* etc. Because Linux is free and open-source, it allows end-users to freely alter, utilize, distribute, and publish software code.

## 2.1 Why Linux?

Programmers and Developers mostly use Linux[2] for Coding and learning Computer Science. Linux[3] is preferred over other OS because -

1. Linux tends to be highly reliable and secure system than any other OS -
   (i) As it is open source, huge number of programmers and developers are fixing the issues and reviewing the code, and hence it has lesser security flaws.
   (ii) You don't need antivirus to protect from malware or viruses because of its security.
   (iii) Linux has more privacy. Windows collects some user data but in Linux, you can always look into its source code and literally see everything transmitting from your system.
2. It is free to use, Open Source and Developer friendly
   (i) It is freely available, saves money on licenses and software.
   (ii) It has powerful and helpful software managers like apt, rpm , dpkg, synaptic etc that make installation process easier.
   (iii) A lot of handy useful and Powerful programming tools comes pre-installed with Linux like grep, wget, cron etc that makes it developer friendly.
3. Task Automation and Command Line Interface
   (i) Linux offers several shells such as **Bash**, SH, Korn, ZSH, Fish through which users can automate thier tasks easily.
   (ii) The shell can not perform its magic until the end-user types some commands in the command-line interface (CLI) through the Linux Terminal.
   (iii) Linux terminal is kind of the *heart of Linux.* It is very efficient and faster.

## 2.2 Some other Features in Linux

Linux has many different features unique to it. Such as a powerful editor known as **Vim**, task automating shells such as **Bash**, text editing streams like **SED** and **AWK**, version control from Command line using **Git** and many more. These features come very handy in use for any programmer and thus programmers should know about these and be well-versed in thier usage. Let's see more about some of them.

# 3 Bash Scripting

The **Linux Bash**[4], also known as 'Bourne-again Shell' is a command language interpreter for the Linux based system. It is a replacement of Bourne shell (sh). It was developed under the GNU Project and written by Brian Fox. The Linux/Unix shell allows us to interact with the Linux system through the commands. It let us invoke an executable file to create a running process.
The Bash is a command language interpreter as well as a programming language. It supports variables, functions, and flow control, like other programming languages. It can also read and execute the commands from a file, which is called a **Shell Script**[5]. Some key features of Bash are Shell syntax, Shell commands, Shell functions, Shell parameters, Redirections, Command executions and Shell scripts.

## 3.1 What is Shell & Scripting?

**Shell**

If we are a new Linux user, and we open the terminal, it is assumed that we are well confused as to what to do with it. Here the Shell comes in the role. The terminal contains the shell; it allows us to execute the commands to interact with the system. We can perform various operations such as store and retrieve data, process information, and various other simple as well as complex tasks.

**Scripting**

Suppose we want to run a series of commands many times to perform a task. Instead of doing the same process many times, Linux supports a feature called scripting that allows to execute more than one task at once. So, it is good to define a script rather than performing repetitive tasks. Script is a file that consists of commands that we want to run. To define a script, we create a file with a .sh extension

## 3.2 More Features of Bash

Well, not only basic commands, we can do many more things using a bash script. Just think about it, Bash can perform those tasks also that a programming language like C++ does it can also execute the commands that we use to write on terminal using **CLI**. A simple example can be that Suppose you are working on a C++ code that involves linking and compilation of several object modules and command line arguements. Now, instead of comipiling all those cpp files to make object modules and then linking them, you can just make a Bash script that will do it for you in a single run.
Also, many other Linux features like SED, AWK, Git, etc can be used within a Bash Script when you are using them for a more complex task.

# 4 Task Automation using Bash

## 4.1 Background

Zomato food delivery needs a persistent storage which can store the information of their customers along with their order details.They need to store Customer's first name and Order ID.Along with this information they also need to keep track of time at which this data is stored in persistent storage.You need to create such storage using files.
You need to make a script that will can perform following tasks -

1. **storeInfo**: Create an entry for <CustomerName, Order ID> pair in storage along with the time and date at which the order was placed.
2. **displayInfo**: Display information of Customer Name and Order ID on terminal.
3. **getOrderID**: Display the Order ID associated with a given Customer Name

## 4.2 Solution with Bash Code

If we would be using any other simple Document for storing this data, it would have been so problematic to keep track the date and time whenever we insert a new entry into our storage and also for finding a particular entry in the storage. But using a Bash Script, we can automate this task easily.

### Algorithm

The first two functions implementation is quite simple and normal, but the last one is a bit tricky. So, let's have a look on the algorithm that can be used for getOrderID funcrion followed by the complete Bash Script code file named as `script.sh` -

**Algorithm 1:** Algorithm for getOrderID function

```
   Input : Name of the text file (fileName) and Name of customer (name) as
           command line arguement
   Output: Order IDs and details of order by given customer
 1 Function getOrderID() :
 2     $N \leftarrow$ numberofOccurence(name,fileName) ;      /* by using grep | wc */
 3     $x \leftarrow$ listofOccurence(name,fileName) ;        /* by using grep command */
 4     $j \leftarrow 0$;
 5     for $i$ in $x$ do
 6         if $j \bmod 2 \neq 0$ then
 7             print i ;         /* only IDs of given name will be printed */
 8         end if
 9         $j \leftarrow j + 1$ ;     /* because name & ID are stored alternatively */
10     end for
11     print N;
12 end Function
```

Following is the implementation of the solution using Bash -

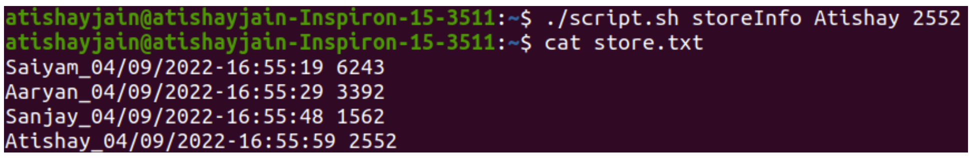
**Listing 1: Bash Script**

```
#!/bin/bash
now="$(date+'%d/%m/%Y')"         #'date' returns today's Date
time=$(date+"%T")
name=$2
underScore="_"
Name="$name$underScore"
id=$3
function storeInfo(){           #Store info of an Order entry
    echo ${name}_${now}-${time} ${id} >> store.txt
}
function displayInfo(){         #Display all Order entries info
    echo Name OrderID
    cat store.txt
}
function getOrderID(){          #Get Order info of a given Name
    echo "OrderID's found are:"
    number=$(tr ' ' '\n' < store.txt | grep $Name | wc -l)
    found=$(grep $Name store.txt)
    j=0
    for i in $found
    do
        if [ $(expr $j % 2) != "0" ]; then
            echo $i
        fi
        j=$(($j+1))
    done
    echo "$name" ordered "$number" times
}
"$@"
```

We can run the above script for performing the required tasks -

1. **storeInfo**

   - For storing an Order ID pair, let's say "Atishay" "2552", we can run the above script as - `./script.sh storeInfo Atishay 2552`



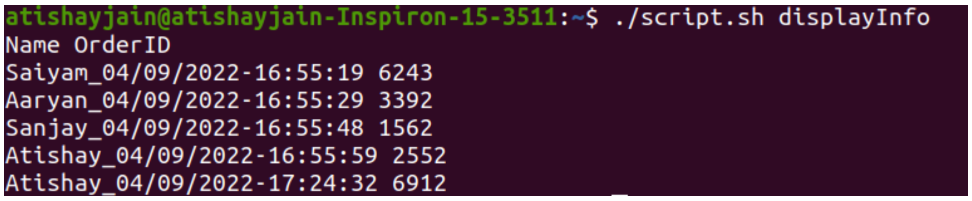
```
atishayjain@atishayjain-Inspiron-15-3511:~$ ./script.sh storeInfo Atishay 2552
atishayjain@atishayjain-Inspiron-15-3511:~$ cat store.txt
Saiyam_04/09/2022-16:55:19 6243
Aaryan_04/09/2022-16:55:29 3392
Sanjay_04/09/2022-16:55:48 1562
Atishay_04/09/2022-16:55:59 2552
```

Figure 1: `cat` command is used here to display content of a file

   - The storeInfo function
     - echoes the string of name, OrderID and other data using `echo` command
     - and redirects it to a storage file `store.txt` using `>>` operator

2. **displayInfo**

   - For displaying all the Order ID's stored in our storage, run the script as - `./script.sh displayInfo`
   - The displayInfo function
     - echoes the string "Name OrderID"
     - uses `cat` command to display store.txt

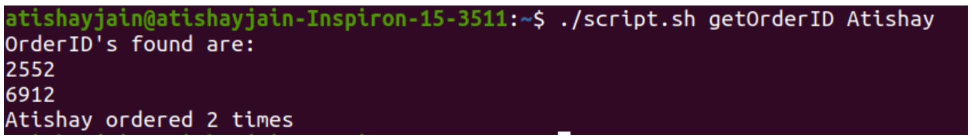


```
atishayjain@atishayjain-Inspiron-15-3511:~$ ./script.sh displayInfo
Name OrderID
Saiyam_04/09/2022-16:55:19 6243
Aaryan_04/09/2022-16:55:29 3392
Sanjay_04/09/2022-16:55:48 1562
Atishay_04/09/2022-16:55:59 2552
Atishay_04/09/2022-17:24:32 6912
```

Figure 2: combination of above two statements results in such a display

3. **getOrderID**

   - To get the Order ID's corresponding to a given name, let's say "Atishay", run the script as `./script.sh getOrderID Atishay`



```
atishayjain@atishayjain-Inspiron-15-3511:~$ ./script.sh getOrderID Atishay
OrderID's found are:
2552
6912
Atishay ordered 2 times
```

Figure 3: It displays all the Order IDs and number of orders by Atishay

   - The getOrderID function uses
     - `echo` command at appropriate places to print some text
     - `grep` command which is used to search for a string in a file
     - `wc` command which is used for finding word count, line count, character count etc
     - `tr`[6] command which is used to translate/delete characters from input and writes the result to output
     - | symbol, that is used for piping. Piping means using output of previous command as input for next command
     - and a for loop using which it finds and prints the order ID and number of orders after looping through the whole storage file

More information about various such Linux Commands and Bash can be found here.

# 5 Sed

**Sed**[7] is the ultimate **S**tream **ed**itor, which can be used to perform lots of functions on file like searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX[8] is for substitution or for find and replace. By using SED you can edit files even without opening them, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it.

- SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution).
- SED command in unix supports regular expression (*Regex*) which allows it perform complex pattern matching.

## 5.1 Background

To understand the power of Sed, let's analyze a situation. Suppose we have a list of emails and we want to check the emails that are valid in this list. Manual checking for a large number of such emails will be very complicated and will take a lot of time and inefficiency. Sed can do this very fast and efficiently.

## 5.2 Work Done

We will use Sed command inside a Bash Script for validating email addresses. The code is following -

**Listing 2: Using Sed inside a Bash Script**

```bash
#!/bin/bash

# Run the script as : ./q1.sh <filename>
# where <filename> is the name of file containing
# the list of many email addresses

sed -n -E -e '/^[a-z0-9A-Z_\!%\+\#\&\$=\?\^~`\/\-]+[\.]?[a-z0-9A-Z_\!%\+\#\&\$=\?\^~`\/\-]+[@]{1}[a-z]+[\.]{1}[a-z]+$/p' -e '/^[a-z0-9A-Z_\!%\+\#\&\$=\?\^~`\/\-]+[\.]?[a-z0-9A-Z_\!%\+\#\&\$=\?\^~`\/\-]+[@]{1}[a-z]+[\.]{1}[a-z]+[\.]{1}[a-z]+$/p' < $1 > valid.txt

# The validated correct email addresses will be printed
# inside a new file named ``valid.txt"
```

The above code uses Regular Expressions and checks each email address. If the expression matches, then it prints those valid emails into a new file. `-E` tag activates the Regex expressions. `/p` at the end is used for printing in Sed, `-n` and `-e` tags are used for printing only when expression matches and using multiple commands in sed respectively.

# 6 Awk

**Awk**[10] is a scripting language, abbreviated from the name of its developers – **A**ho, **W**einberger, and **K**ernighan. It is used for manipulating data and generating reports. The awk command programming language requires no compiling and allows the user to use variables, numeric functions, string functions, and logical operators. [9]
Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. Awk searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions.

## 6.1 What can awk do?

| | |
|---|---|
| Awk Operations[11] | Scans a file line by line |
| | Splits each input line into fields |
| | Compares input line/fields to pattern |
| | Performs action(s) on matched lines |
| Programming Constructs | Format output lines |
| | Arithmetic and string operations |
| | Conditionals and loops |
| Useful For | Transform data files |
| | Produce formatted reports |

Table 2: Uses and applications of awk

## 6.2 An example using Awk

How about a simple challenge! Do you know about a tool called 'wc'? It helps you count the no. of lines, no. of words, characters and even bytes of a text file.
Can you design your own version of 'wc' without using 'wc' itself? Let us trivialize the problem a little. Write a bash script to take a filename as an input argument and calculate the no. of paragraphs in it. The script should be able to take a flag `-para` as arguement and display the number of paragraphs in it. Let's solve this problem and look at code of it using awk!

## 6.3 Algorithm

**Algorithm 2:** Algorithm for counting number of paragraphs

```
Input  : Name of the text file and -para flag as command line arguement
Output: Number of paragraphs in the text file
1  $emptyline \leftarrow 0$;
2  for line in textfile do
3      if Not end of file then
4          if $line.(words) = 0$ then
5              $emptyline = emptyline + 1$;
6          end if
7      end if
8  end for
9  $para \leftarrow emptyline + 1$;
10 print para;
```

## 6.4 Code

**Listing 3: Bash code using awk**

```
#!/bin/bash
 flag=$2
 if [[ $flag == "-paras" ]];
then
sed '/^$/N;/^\n$/D' $1 | awk '!NF {para += 1} END {
     if(NF == 0) print para " paragraphs"
     else print para+1 " paragraphs"
     }'
 fi
```

The above code will print the number of paragraphs of a given text file. It uses awk inside a Bash Script. The keyword `NF` in awk contains the number of words in a line. Awk divides the text file into records and fields, normally using record separator as End line character and field separator as whitespace. So, `NF` shows the number of words in a record. The code is calculating empty line using condition that `NF = 0` for empty line, and the assumption that two paragraphs have empty line between them leads to finding number of paragraphs.

## References

[1] Wikipedia. (2022) Linux. Last Edited: 05-09-2022. [Online]. Available: https://en.wikipedia.org/wiki/Linux

[2] mrNayaNi. (2021) Why (many) programmers prefer linux over windows. Published: 10-01-2021. [Online]. Available: https://www.cybrary.it/blog/0p3n/programmers-prefer-linux-windows/

[3] S. K. Soumik. (2021) Reasons why linux is better than windows. Published: 21-01-2021. [Online]. Available: https://medium.com/swlh/13-reasons-why-linux-is-better-than-windows-6fa304454ae

[4] Wikipedia. (2022) Bash (unix shell). Last Edited: 01-09-2022. [Online]. Available: https://en.wikipedia.org/wiki/Bash_(Unix_shell)

[5] JavaTpoint. Linux bash. [Online]. Available: https://www.javatpoint.com/linux-bash

[6] B. Marijan. (2022) Linux tr command with examples. Published : 10-05-2022. [Online]. Available: https://phoenixnap.com/kb/linux-tr

[7] B. Barnett. (1994) Sed - an introduction and tutorial. Last Modified: Mon Dec 7 10:13:59 2020. [Online]. Available: https://www.grymoire.com/Unix/Sed.html

[8] A. Rajput and M. A. (GeeksforGeeks). Sed command in linux/unix with examples. Last Updated: 21 Dec, 2021. [Online]. Available: https://www.geeksforgeeks.org/sed-command-in-linux-unix-with-examples/

[9] Wikipedia, the free encyclopedia, “Awk,” [Online; accessed September 5 , 2022]. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/d/d5/The-AWK-Programming-Language.svg

[10] Wikipedia. Awk. Last Edited: 25 August 2022, at 08:50 (UTC). [Online]. Available: https://en.wikipedia.org/wiki/AWK

[11] A. Goyal and P. N. (GeeksforGeeks). Awk command in unix/linux with examples. Last Updated : 17 Jun, 2022. [Online]. Available: https://www.geeksforgeeks.org/awk-command-unixlinux-examples/