MINOR-1
PROJECT REPORT on

ACTIVITY FETCHER: USER ACTIVITY DETECTION SYSTEM

Submitted By:

Name	Roll No	Branch	SAP-id
Himanshu Jain	R110216074	CSE CCVT	500052189
Ishu Khanchi	R110216080	CSE CCVT	500053026
Nihal Agarwal	R110216103	CSE CCVT	500052549
Pawas Seth	R110216109	CSE CCVT	500053003

Under the guidance of

Dr Hitesh Kumar Sharma
Assistant Professor (Selection Grade)
Department of Cybernetics

School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
Dehradun-248007, 2018-2019

Approved By:

(Dr Hitesh Kumar Sharma)
Project Guide

(Dr Amit Agarwal)
Department Head

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide Dr. Hitesh Kumar Sharma, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department, Dr. Amit Agarwal, for his great support in doing our project in Area at CIT.

We are also grateful to Dr. Manish Prateek Professor and Director SoCSE and Dr. Kamal Bansal Dean CoES, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to give special mention and our deep gratitude and regards to Mr. Shailendra Srivastava (BFSI), University of petroleum and Energy studies for his guidance and showing us the right direction to carry on the project.

We would like to thank all our friends for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our parents who have shown us this world and for every support they have given us.

Name	HIMANSHU JAIN	ISHU KHANCHI	NIHAL AGARWAL	PAWAS SETH
Roll No.	R110216074	R110216080	R110216103	R110216109

TABLE OF CONTENTS

S.No. Contents

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

End Semester Report (2018-19)

1 Project Title

ACTIVITY FETCHER:USER ACTIVITY DETECTION SYSTEM

2 Abstract

These days, security is to be given highest priority, and that too without compromising user's privacy, but still, this is a matter of concern for a lot of users and administrators. Many software's/installers came into market but each one has its own issues and disadvantages. Many suspicious user activities can compromise security across the network and will lead to data breaches, so finding the root cause of the breach or any other problem regarding file system. Manual Analysis increases the time to solve the threats. Keeping this in mind the ultimate need of a tracer to track down the activities of any third party given access by the user/administrator, this project is a utility tool. This utility tool is made to monitor and trace the activities done on the user's system by any third party and keeping the logs of those activities in the log files. We have used the concept of different Linux commands, and have implemented their functionality in C programming language. We have integrated all the functionality to make the application user friendly and user and administrator can use for different purposes like knowing the uptime of the system or for security concerns and many more. This is also used for monitoring the activities of the user. Associations are established between the states of certain computer system parameters and specific activities. Strategies and frameworks are accommodated catching utilization information from a client PC, preparing a subset of such information to shape yield, and offering access to perspectives of such yield, for example, to help an organization's administration in checking PC use in a workplace. A system and method for monitoring computer usage is disclosed. A computer operator specifies discrete moments of a computer's usage at which screen captures are executed and saved to a log. By constantly recording the logs, users will be able to determine/predict strategize security failovers and appropriately accommodate the precautionary measures. For e.g.: If any third person given access to the system, removes any files or make sudden changes without user's permission, user can track that from the records stored in log files and can take actions against him.

Keywords: Tracing and monitoring system, Monitoring network, Log files management.

3 Introduction

Monitoring and administering have become a vital part in the technological era. The main aim of this project is to provide users with the information about the network and device logs to keep track on the person using the system and to allow the administrator to take necessary precautions before any damage to his files or system. There are tools existing which helps in tracing down various activities done on the system or network. The project will be on collecting the data and then recording that in the log files to allow administrator to monitor his system on or over the network and take precautionary steps by implementing Linux functionalities through C.

Observing your Linux framework is fundamental to have the capacity to enhance its execution, find the wellspring of an issue and take more focused on remedial activities. As is dependably the situation with Linux, there are many devices and a wide range of ways you can use to screen diverse parts of your framework's execution. By implementing user activity monitoring, enterprises can more readily identify suspicious behavior and mitigate risks before they result in data breaches, or at least in time to minimize damages. Sometimes called user activity tracking, user activity monitoring is a form of surveillance, but serves as a proactive review of end user activity to determine misuse of access privileges or data protection policies either through ignorance or malicious intent. Taking in the account the safety of itself and high reliability features, we need to do design more complete security monitoring system, which combines various log files accessing, various commands to monitor the activities of the user, thus able to track and monitor the user activities. Development of information security to information security has brought new challenges. System activity monitoring system is an effective information security mechanism, and system monitoring is an important area of system applications, the demand for applications is also increasing with the development and the network technology, has created a safety monitoring system hardware framework.

With the size of the enterprise network is growing, more and more computer terminals, internal information leakage and security threats emerging security incidents and daily management to bring a lot of pressure. Business users need a tracking system to address these security issues.

A technique and framework for checking and following the exercises of a client of a PC. Affiliations are set up between the conditions of certain PC framework parameters and explicit exercises. The present movement of the client is then definite by the framework dependent on the present condition of the PC framework. As the condition of the PC framework changes, changes in the client's action are checked and followed. The exercises are followed related to time in order to record the time spent on every movement.

3.1 The Benefits of User Activity Monitoring

Any level of monitoring can accumulate large amounts of data. The goal of any user activity monitoring program should be to find and filter out actionable information that’s valuable in data protection efforts. With effective processes in place, you can immediately detect and investigate suspicious user activity. You can also find out if users are uploading sensitive data to public clouds, utilizing non-approved services and applications, or engaging in any other type of risky activity while using the company network or resources. User activity monitoring tools are also helpful in ensuring that employees do not take any of your company's confidential information when they are leaving the company.

In order to make the data collected by user activity monitoring solutions as useful as possible, that data must be analysed for several items, including:

- Associated risk
- Defined policies
- Time of day
- Identity context

It also helps to have real-time identification along with detailed reporting of historical activity. Questions to answer are: Who did what, when and where? User activity monitoring helps to identify abuse to help reduce the risk of inappropriate actions that can lead to malware infections or data breaches. It also helps to decrease the cost of compliance, while offering intelligence needed to improve security measures.

User activity alerting serves the purpose of notifying whoever operates the UAM solution to a mishap or misstep concerning company information. Real-time alerting enables the console administrator to be notified the moment an error or intrusion occurs. Alerts are aggregated for each user to provide a user risk profile and threat ranking. Alerting is customizable based on combinations of users, actions, time, location, and access method. Alerts can be triggered simply such as opening an application or entering a certain keyword or web address. Alerts can also be customized based on user actions within an application, such as deleting or creating a user and executing specific commands.

4 Literature Review

Once you've determined a user of suspicious activities across the network or on the system, you can do an in-depth search to identify and determine the user's historical activity. Additionally, Log files help you define correlation rules centered around this specific user so that you can automate alerting to all events triggered by the user activity. Every tracing and monitoring tool provide statistical information, but this tool also provide an option to record the data in log files along with the performance monitoring of system and network. This tool can be extended for many other devices and any kind of machine logs.

- [6] Log & Event Manager features an in-memory correlation engine that provides security incident awareness in real-time. You can choose from hundreds of built-in rule templates and customize them to detect and respond to suspicious user activities, such as adding or removing users from admin groups, accessing a business-critical server after office hours, and more. Some of the corrective actions upon detecting these anomalies include blocking IP addresses, sending alert pop-ups or emails, logging off the user, restarting or shutting down the source machines, etc.

- [7] User activity monitoring (UAM) solutions are software tools that monitor and track end user behavior on devices, networks, and other company-owned IT resources. Many organizations implement user activity monitoring tools to help detect and stop insider threats, whether unintentional or with malicious intent. The range of monitoring and methods utilized depends on the objectives of the company. By implementing user activity monitoring, enterprises can more readily identify suspicious behavior and mitigate risks before they result in data breaches, or at least in time to minimize damages. Sometimes called user activity tracking, user activity monitoring is a form of surveillance, but serves as a proactive review of end user activity to determine misuse of access privileges or data protection policies either through ignorance or malicious intent.

- In paper [8] by Janet Yoo, Kian-Tat Lim, Stanley Ben Wong and Elliott Yasnokvsky. This gives us the info of a traffic monitor providing statistics of traffic using an activity input for receiving data related to activity on a server system. Events being monitored are binned by topic or term, where the terms are associated with categories. The categories can be a hierarchy of categories and subcategories, with terms being in one or more categories. The categorized events include page views and search requests and the results might be normalized over a field of events and a result output for outputting results of the normalizer as the statistical analyses of traffic.

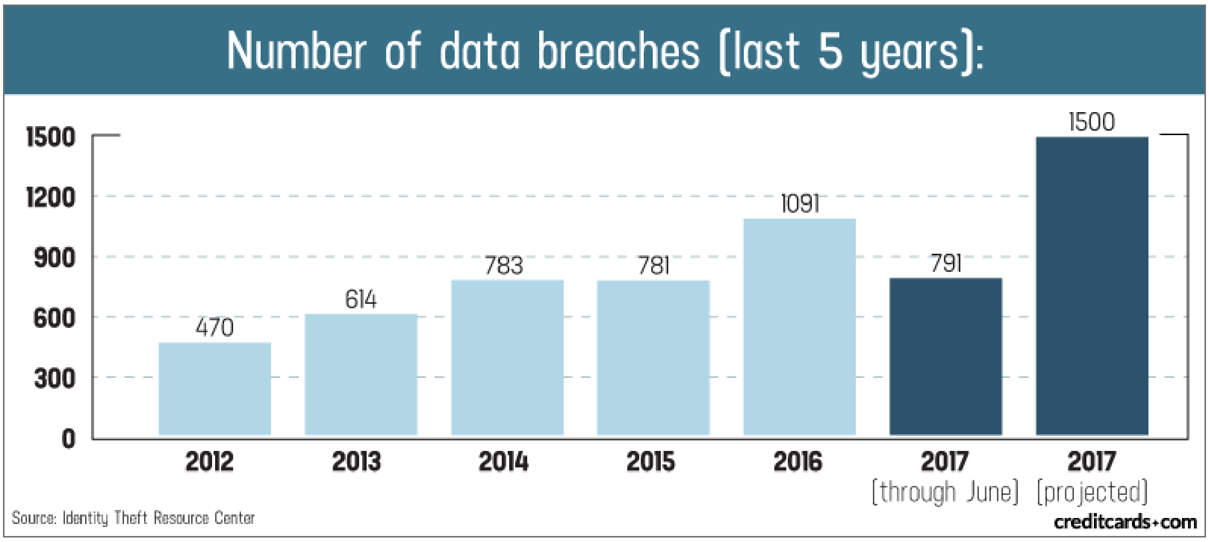
- In paper [9] by Eric Anderholm and David Losen, in this, Invention relates to the field of monitoring system usage, and more particularly to the field of using software to monitor user, application and device behavior and events. The present invention relates to the use of systems to monitor user, application and device behavior and events, including, without limitation, to monitor productivity and to monitor compliance with workplace policies and regulations. In embodiments, the systems may be used to capture usage data from a user computer, process such data to form, and offer access to, selective views of such output, such as to assist a company's management in monitoring computer usage in a work environment. In embodiments, the output may be processed and viewed according to software application, device, or specified user. The output, or a report generated from the output, may be accessible in differing degrees to individuals having appropriate levels of permission.

- [10] In the field of information security, user activity monitoring (UAM) is the monitoring and recording of user actions. UAM captures user actions, including the use of applications, windows opened, system commands executed, check boxes clicked, text entered/edited, URLs visited and nearly every other on-screen event to protect data by ensuring that employees and contractors are staying within their assigned tasks, and posing no risk to the organization. UAM software can deliver video-like playback of user activity and process the videos into user activity logs that keep step-by-step records of user actions that can be searched and analysed to investigate any out-of-scope activities.

- [11] File activity monitoring products are designed to monitor the patterns of legitimate users accessing enterprise file stores and alert security administrators to unusual activity. FAM is designed to go above and beyond the access control and logging capabilities built-in to operating systems, providing a usable way to perform both proactive and reactive security monitoring. FAM products typically integrate with other products in your environment, including Active Directory/LDAP information stores, DLP and other elements of your security infrastructure. For example, the FAM might leverage Active Directory groups to assign role-based policies to users without requiring the roles to be populated on the FAM itself.

- [12] Computer and network surveillance is the monitoring of computer activity and data stored on a hard drive, or data being transferred over computer networks such as the Internet. The monitoring is often carried out covertly and may be completed by governments, corporations, criminal organizations, or individuals. It may or may not be legal and may or may not require authorization from a court or other independent government agencies. Computer and network surveillance programs are widespread today and almost all Internet traffic can be monitored.

- In paper [13] by Gene Goykhman, the user's activities can be tracked in connection with several variables, the most important of which is likely time. By tracking time in connection with a user's activities on a computer, one can monitor how much time is spent on activities. This can be essential information for project management, project assessments, efficiency analysis, billing and organizational management. A user's activities can also be tracked in connection with other variables such as network or processor loading.



- [14] By utilizing the framework and technique for the unveiled development, a record of a client's action or inertia is made that, in addition to being highly simple to evaluate, is an irrefutable account of the user's computer usage. Varieties of the framework and technique enable the administrator to coordinate checking occasions toward on the web or disconnected exercises. Finally, we have found the way to tackle the se One can access the services further through various user interface tools. Tool is supported by Linux and any log file can be read by the administrator.

5 Problem Statement

Suspicious user activity can compromise security across the network, leading to a data breach. Finding the root cause of an incident, and tracing it to a suspected user, becomes difficult without comprehensive audit trails. Non-centralized and manual log analysis increases the time it takes to remediate threats and prevent business downtime. Thus, there’s a need to find a system that can trace all these unwanted activities and user can take precautions accordingly.

6 Objectives

The main objective of this project is:

- Tracing down users’ activities on the system.
- Maintaining and accessing the log files that store the session time.

7 System Requirements (Software/Hardware)

Software: GUI Toolkit, Ubuntu OS, gcc compiler
Hardware: 4GB RAM, 512GB Hard Disk, 2.3GHz processor

8 Algorithm

Step 1) Start

Step 2) There are three modules accessible at present. User enter his choice of the functionality he wants to perform.

Step 3) Using nested if-else to get the user choice in input

```
{
        case p: To view all the active and user processes.
        case c: To display the timespot.
        case l: To display the files list in the directory.
        case u: To display the running time.
        case o: To access the log files of the system in ubuntu.

}
```

Step 4) If case o:

a. Start
b. Create a variable of type "FILE*".
c. Open the log files that you want to access using the "fopen" function and assign the "file" to the variable.
d. Find the size of the file:sz using fseek and ftell gives the size as the pointer traces till the end of the file.
e. Initialize buffer size dynamically using malloc().
f. This will store the characters from the file to be read having size: (sz*sizeof(char))
g. Take the pointer back to first character rewind (infile)
h. Check to make sure the file was successfully opened by checking to see if the variable == NULL.
i. Use the fread() function to read from the buffer that reads character one by one till i<sz.
j. fclose(infile) closes the file
k. End

Step 5) If case p:

a. START.
b. Input choice from user.
c. if choice = d
    - Execute system command to view all active and user processes
      System(ps aux)
d. if choice = s
    - Input the directed path of the file from user.
    - Execute system command to store all active user processes in an external file
      System(ps aux > {filepath})
e. else go to STEP 2

f. END

Step 6) If case l:

a. Get user input. After you input to the command line, shell reads your input by using getline() function.

b. Check for expansions and alias. Prior to searching for the ls command, shell will check for special characters that need to be expanded. Shell also checks for aliases.

c. Check built-in ls is not an alias, shell will then check whether the command ls is a built-in command. A built-in command is a command that is built into and executed in shell itself.

d. Check PATH shell will look for ls in the environment variable, **PATH**. First, a copy of PATH will be tokenized(separated by :) with each representing the path to a directory. A copy is necessary here because we don't want to alter PATH.

e. The ls file will contain the code to display all he directories.
   - DIR *d;
   - struct dirent *dir;
   - d = opendir(".");
   - if (d) {
   - while ((dir = readdir(d)) != NULL) {
   - printf("%s\n", dir->d_name):}
   - closedir(d); }

f. If file exists: fork and execute program in the child process.
   - Shell(*calling process/parent process*)will call fork() to create a copy of itself(*child process*). The clone/child process will have its own system process ID.
   - The child process then calls execve() to run the user's command,ls. execve() will replace the current(child) process with the program it calls(ls, in this case).
   - Parent process waits until the child competes its execution via wait().

g. After child process terminates, parent process will print prompt to user.

Step 7) If case u:

a. We will use a function: err_exit (const char *format, ...), in which have a variable of va_list and then pass it inside vfprintf() function, then the function ends or error message get displayed. This function is basically for error handling.

b. Now, we have uptime(double *uptime_secs, double *idle_secs) function in which we open the file "/proc/uptime" in read only view either for user or group or others.

```
If (file gets open) {
        If (file descriptor isn't null) {
                If (check that we have 2 arguments and save their data
                to read_buf simultaneously) {
```

```
        else {calls err_exit() }
    else {calls err_exit() }}
else {calls err_exit() }}.
```

This function is for checking the number of arguments in the uptime file.

c. Now comes the loadavg(double *av1, double *av5, double *av15) in which we open the file "/proc/loadavg" in read only view either for user or group or others.

```
If (file gets open) {
    If (file descriptor isn’t null) {
        If (check that we have 3 arguments and save their data
        to read_buf simultaneously) {
        else {calls err_exit() }
    else {calls err_exit() }}
else {calls err_exit() }}.
```

This function is for checking the number of arguments in the loadavg file.

d. We use the pre-defined function time() and use it to get the current time and save it to variable and pass it to localtime() to break down the time to hours, minutes and seconds and save it to out_buff array.

e. Similarly, update the uptime and idle time in the out_buff array.

f. Print the data in the out_buff array; which prints uptime with the percentage utilization of CPU before 1 minute, 5 minutes and 15 minutes.

Step 8) In case c:

a. Firstly, We will define my_strftime(const char *format, struct tm *tmp) in which we use pre-defined function strftime(char *s, size_t max, const char *format, const struct tm *tmp), which gets the time in a format and save it to a variable.

b. We save the current time in tmp by using localtime() which divides the data in particular time format defines manually or defined in the buf array.

c. Print the ctime() , asctime() and strftime() which defines the time in different formats.

Step 9) End.

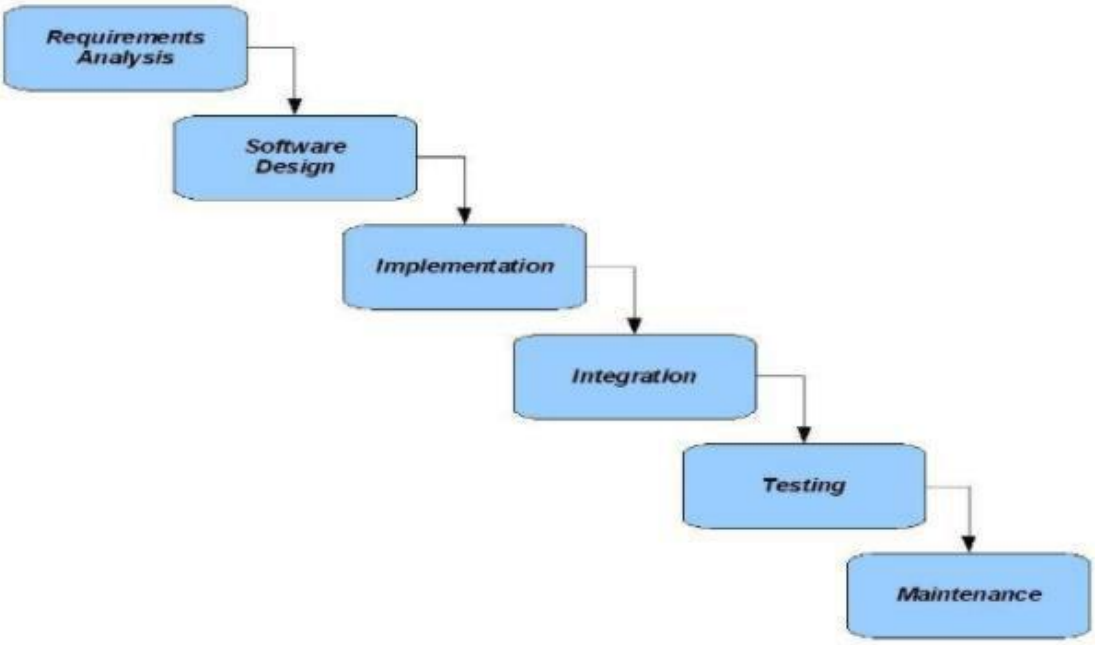
9 Methodology

This project will be implemented using C. Various commands of Linux will be implemented in C to monitor the activities of third party working on the system and doing any changes on the system or on the network by the help of process list. The Process list is the list maintained in a system that keeps the track of every process executed in a device. We will be the accessing the process list to get the session time for the process. Then, this data will be stored in the log files or the database to maintain a record that can be accessed by the administrator. Using this log file, administrator can check the illegal access of the system. Further user interface tools can be used to provide different options to the user like performance monitoring of the network etc.

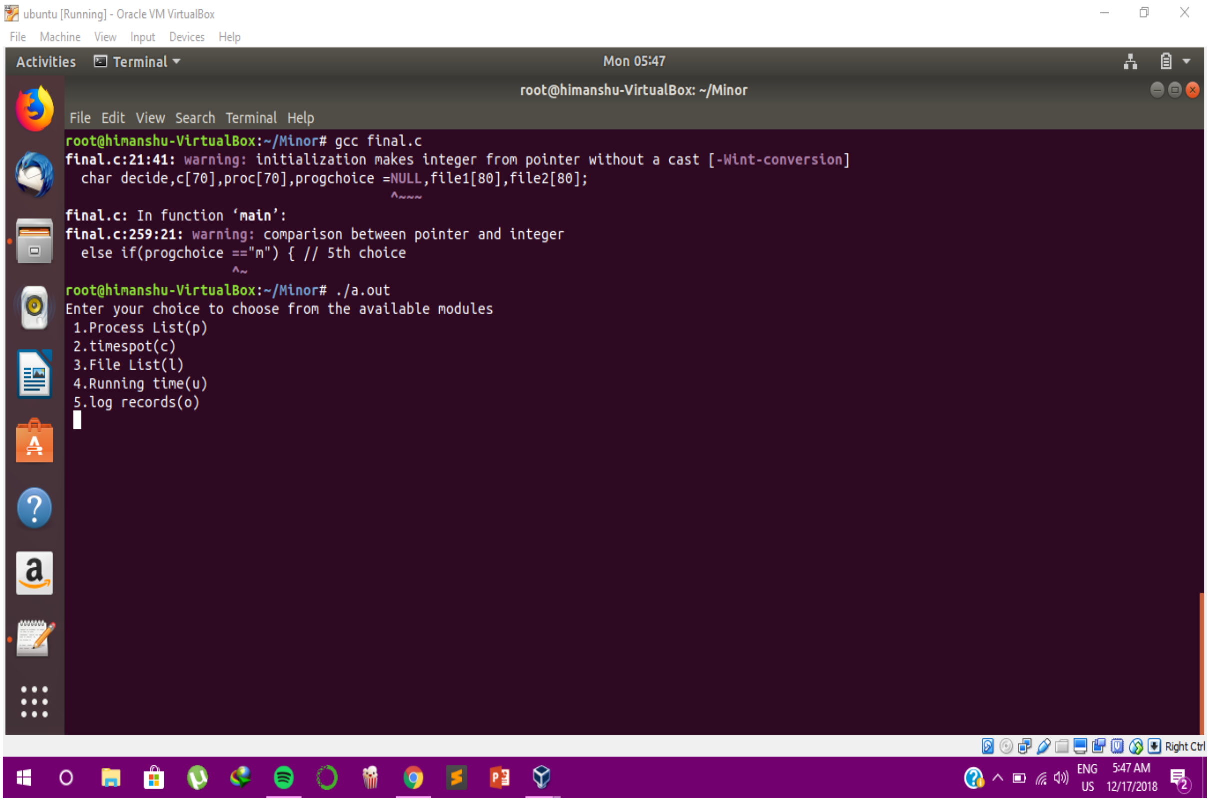
The complete project is divided into 2 modules:

1) Device activity tracer Module
2) Log monitoring module

Fig 3.1: Waterfall model for the project

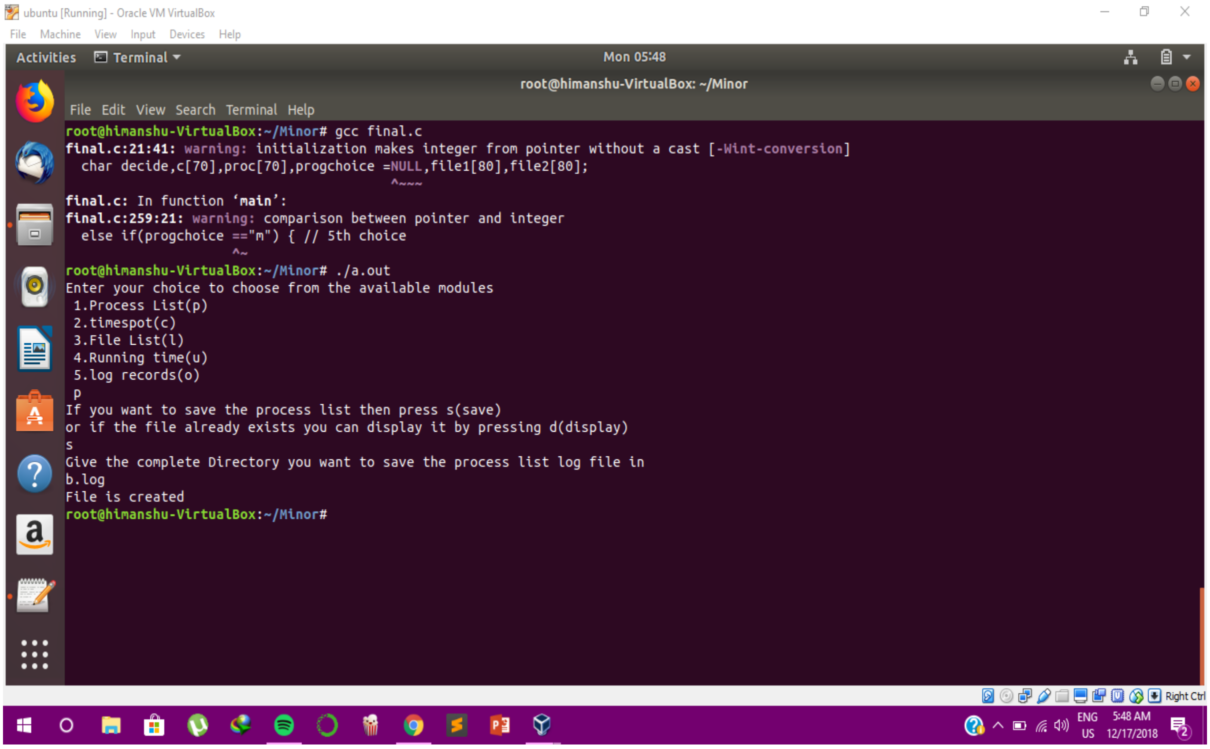


9.1 Implementation and Integration


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:47
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG
US
5:47 AM
12/17/2018

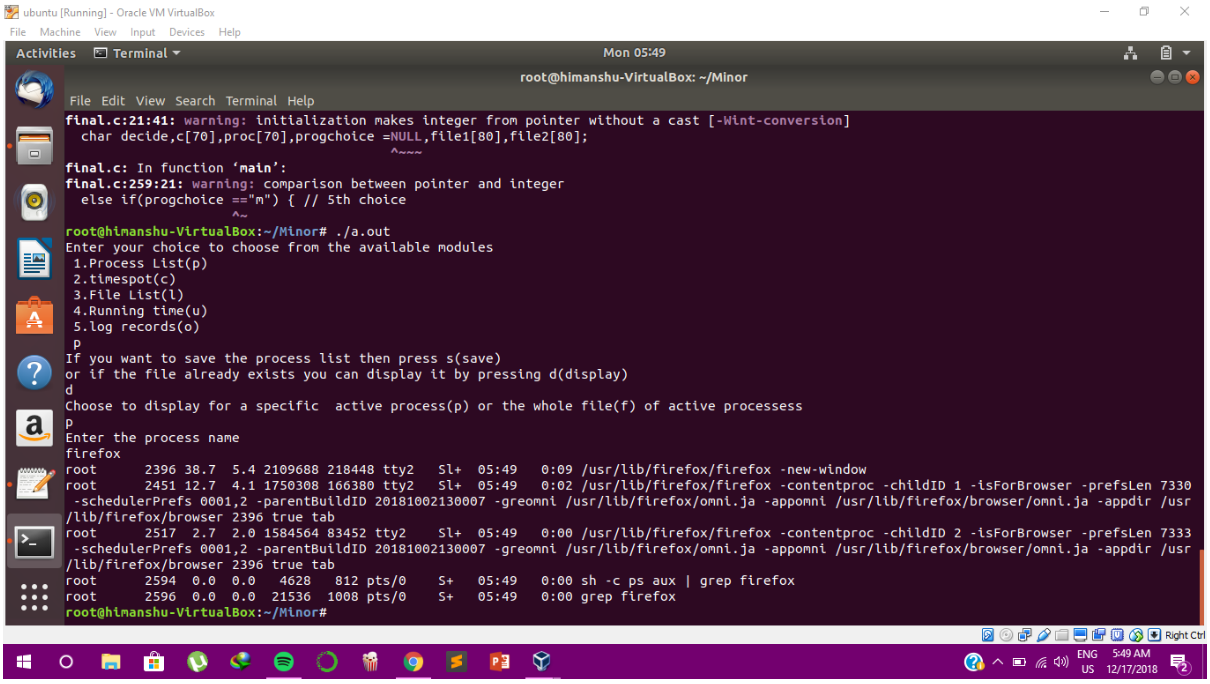
```
root@himanshu-VirtualBox:~/Minor# gcc final.c
final.c:21:41: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
  char decide,c[70],proc[70],progchoice =NULL,file1[80],file2[80];
                                         ^~~~
final.c: In function ‘main’:
final.c:259:21: warning: comparison between pointer and integer
  else if(progchoice =="m") { // 5th choice
                     ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
```

Fig. 9.1.1: User Interface with the various options for the user to select.


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:48
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG
US
5:48 AM
12/17/2018

```
root@himanshu-VirtualBox:~/Minor# gcc final.c
final.c:21:41: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
  char decide,c[70],proc[70],progchoice =NULL,file1[80],file2[80];
                                         ^~~~
final.c: In function ‘main’:
final.c:259:21: warning: comparison between pointer and integer
  else if(progchoice =="m") { // 5th choice
                     ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
 p
If you want to save the process list then press s(save)
or if the file already exists you can display it by pressing d(display)
s
Give the complete Directory you want to save the process list log file in
b.log
File is created
root@himanshu-VirtualBox:~/Minor#
```

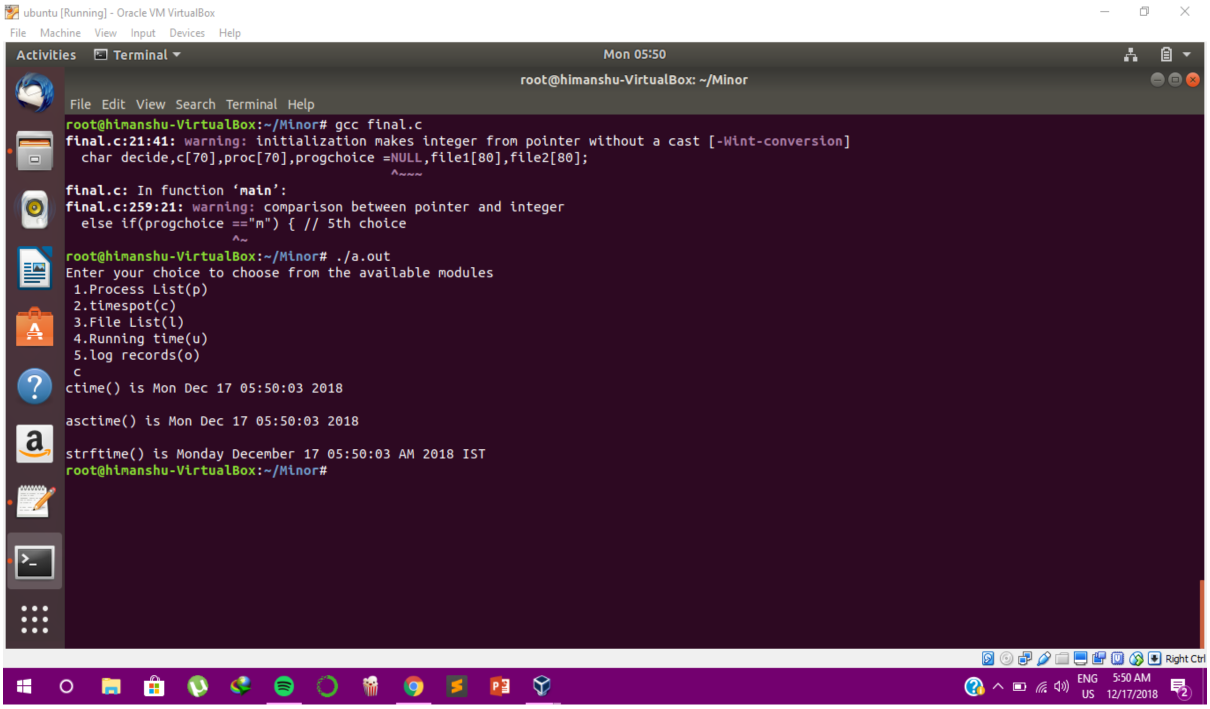
Fig. 9.1.2: Showing the list of the processes that are currently running.



ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:49
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG US
5:49 AM
12/17/2018

```
final.c:21:41: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
  char decide,c[70],proc[70],progchoice =NULL,file1[80],file2[80];
                                         ^~~~
final.c: In function ‘main’:
final.c:259:21: warning: comparison between pointer and integer
  else if(progchoice =="m") { // 5th choice
                     ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
 p
If you want to save the process list then press s(save)
or if the file already exists you can display it by pressing d(display)
d
Choose to display for a specific  active process(p) or the whole file(f) of active processess
p
Enter the process name
firefox
root      2396 38.7  5.4 2109688 218448 tty2   Sl+  05:49   0:09 /usr/lib/firefox/firefox -new-window
root      2451 12.7  4.1 1750308 166380 tty2   Sl+  05:49   0:02 /usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefsLen 7330
 -schedulerPrefs 0001,2 -parentBuildID 20181002130007 -greomni /usr/lib/firefox/omni.ja -appomni /usr/lib/firefox/browser/omni.ja -appdir /usr
/lib/firefox/browser 2396 true tab
root      2517  2.7  2.0 1584564 83452 tty2    Sl+  05:49   0:00 /usr/lib/firefox/firefox -contentproc -childID 2 -isForBrowser -prefsLen 7333
 -schedulerPrefs 0001,2 -parentBuildID 20181002130007 -greomni /usr/lib/firefox/omni.ja -appomni /usr/lib/firefox/browser/omni.ja -appdir /usr
/lib/firefox/browser 2396 true tab
root      2594  0.0  0.0   4628   812 pts/0    S+   05:49   0:00 sh -c ps aux | grep firefox
root      2596  0.0  0.0  21536  1008 pts/0    S+   05:49   0:00 grep firefox
root@himanshu-VirtualBox:~/Minor#
```

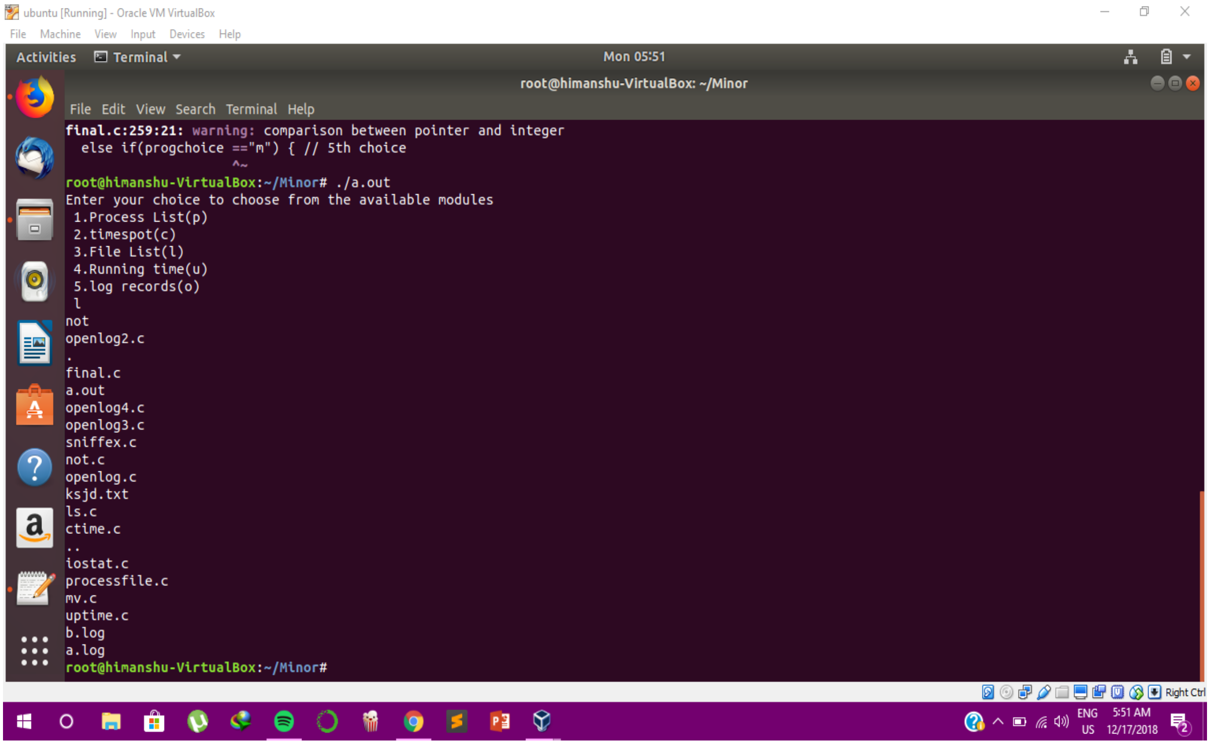
Fig. 9.1.3: Further detailed execution and output of the process list command.



ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:50
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG US
5:50 AM
12/17/2018

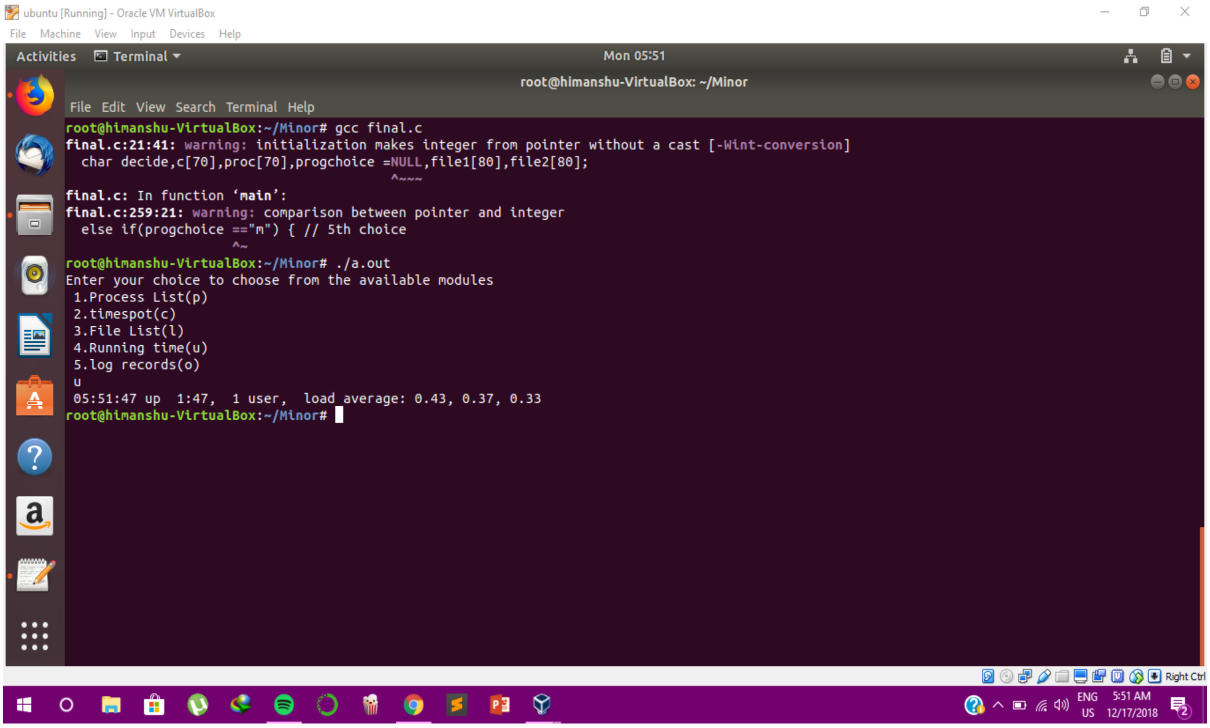
```
root@himanshu-VirtualBox:~/Minor# gcc final.c
final.c:21:41: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
  char decide,c[70],proc[70],progchoice =NULL,file1[80],file2[80];
                                         ^~~~
final.c: In function ‘main’:
final.c:259:21: warning: comparison between pointer and integer
  else if(progchoice =="m") { // 5th choice
                     ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
 c
ctime() is Mon Dec 17 05:50:03 2018

asctime() is Mon Dec 17 05:50:03 2018

strftime() is Monday December 17 05:50:03 AM 2018 IST
root@himanshu-VirtualBox:~/Minor#
```

Fig. 9.1.4: User selecting the timespot command and getting the output.


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:51
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG US
5:51 AM 12/17/2018

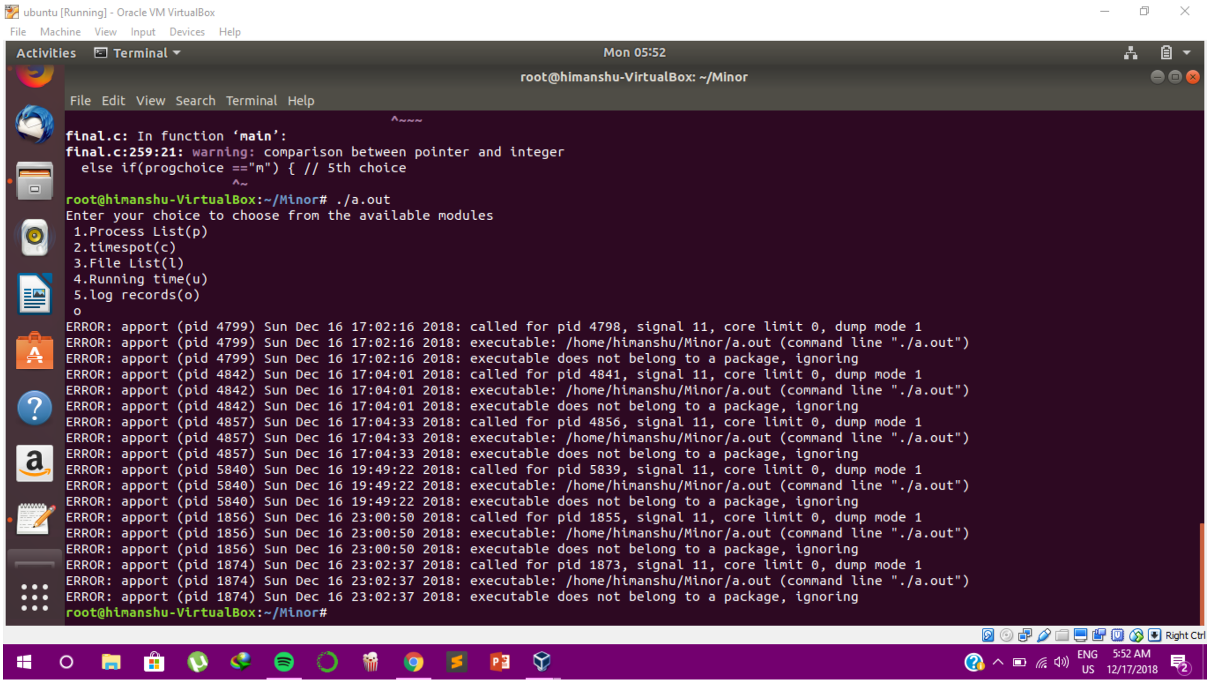
```
final.c:259:21: warning: comparison between pointer and integer
   else if(progchoice =="m") { // 5th choice
                      ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
 l
not
openlog2.c
.
final.c
a.out
openlog4.c
openlog3.c
sniffex.c
not.c
openlog.c
ksjd.txt
ls.c
ctime.c
..
iostat.c
processfile.c
mv.c
uptime.c
b.log
a.log
root@himanshu-VirtualBox:~/Minor#
```

Fig. 9.1.5: User selecting the File list command and getting all the files in the directory as output.


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:51
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG US
5:51 AM 12/17/2018

```
root@himanshu-VirtualBox:~/Minor# gcc final.c
final.c:21:41: warning: initialization makes integer from pointer without a cast [-Wint-conversion]
   char decide,c[70],proc[70],progchoice =NULL,file1[80],file2[80];
                                         ^~~~
final.c: In function 'main':
final.c:259:21: warning: comparison between pointer and integer
   else if(progchoice =="m") { // 5th choice
                      ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
 u
 05:51:47 up  1:47,  1 user,  load average: 0.43, 0.37, 0.33
root@himanshu-VirtualBox:~/Minor#
```

Fig. 9.1.6: Selection of the Running time command and getting the CPU utilization as output.

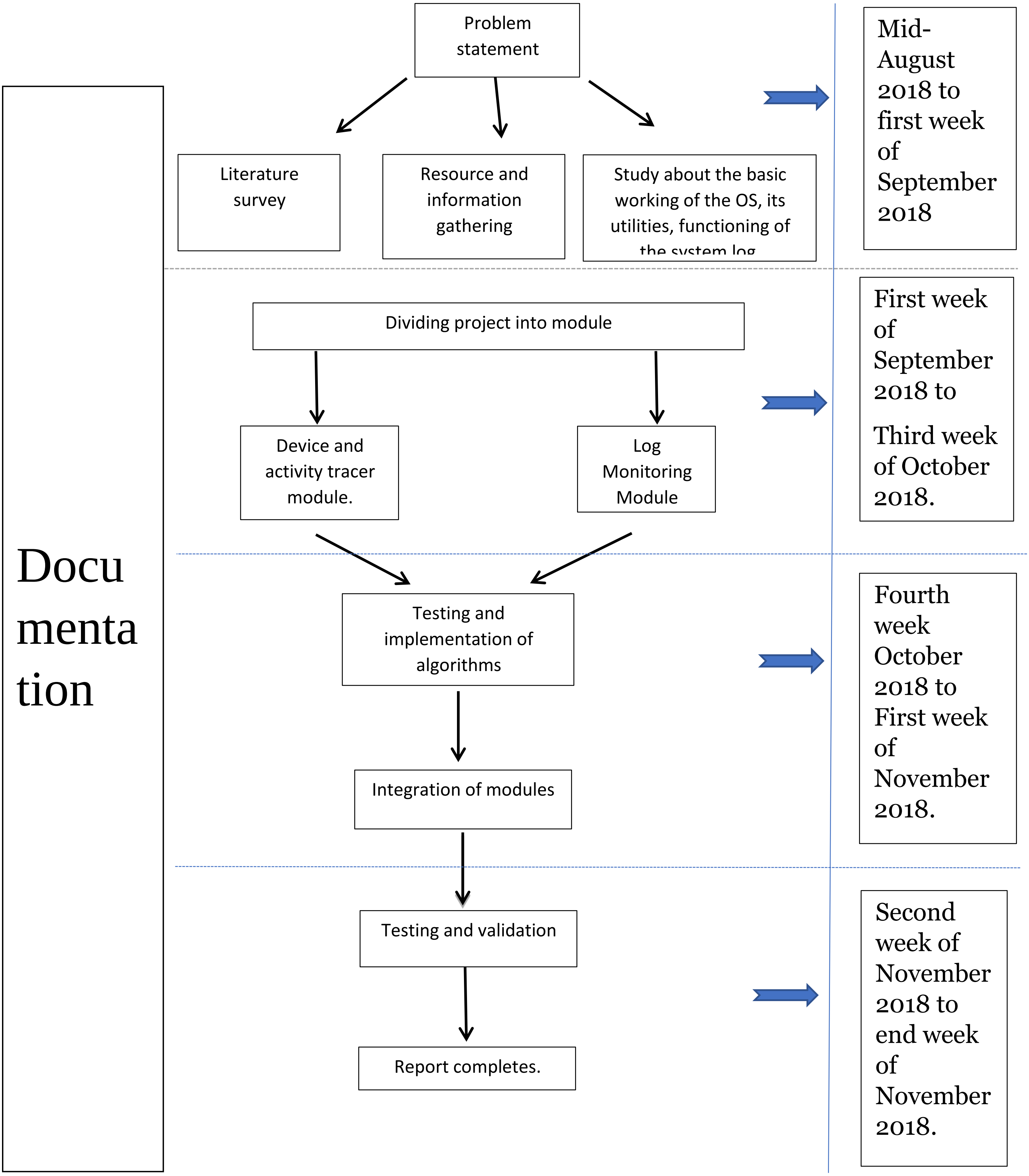


ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 05:52
root@himanshu-VirtualBox: ~/Minor
File Edit View Search Terminal Help
Right Ctrl
ENG
US
5:52 AM
12/17/2018

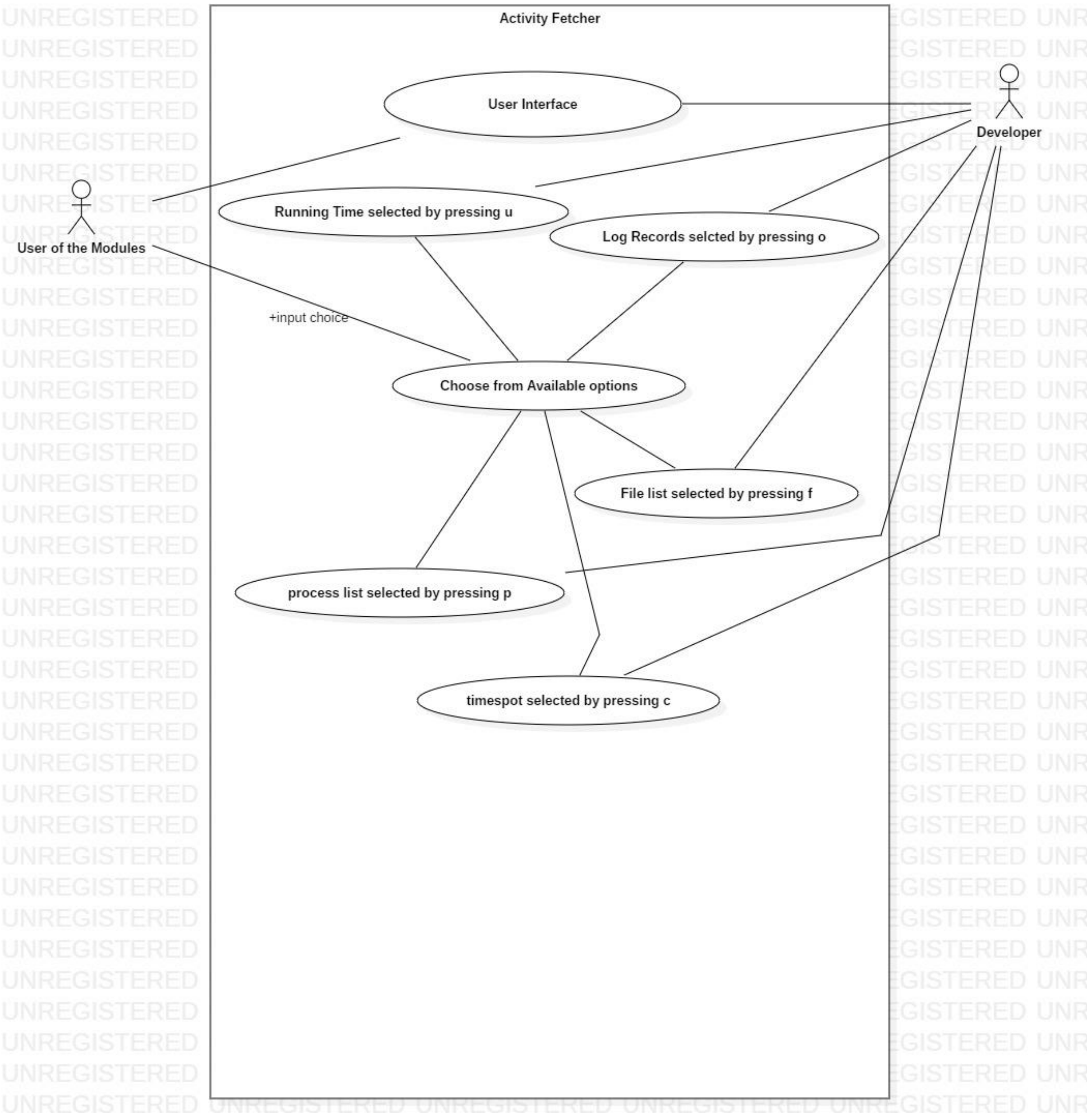
```
                                                                            ^~~~
final.c: In function ‘main’:
final.c:259:21: warning: comparison between pointer and integer
  else if(progchoice =="m") { // 5th choice
                     ^~
root@himanshu-VirtualBox:~/Minor# ./a.out
Enter your choice to choose from the available modules
 1.Process List(p)
 2.timespot(c)
 3.File List(l)
 4.Running time(u)
 5.log records(o)
 o
ERROR: apport (pid 4799) Sun Dec 16 17:02:16 2018: called for pid 4798, signal 11, core limit 0, dump mode 1
ERROR: apport (pid 4799) Sun Dec 16 17:02:16 2018: executable: /home/himanshu/Minor/a.out (command line "./a.out")
ERROR: apport (pid 4799) Sun Dec 16 17:02:16 2018: executable does not belong to a package, ignoring
ERROR: apport (pid 4842) Sun Dec 16 17:04:01 2018: called for pid 4841, signal 11, core limit 0, dump mode 1
ERROR: apport (pid 4842) Sun Dec 16 17:04:01 2018: executable: /home/himanshu/Minor/a.out (command line "./a.out")
ERROR: apport (pid 4842) Sun Dec 16 17:04:01 2018: executable does not belong to a package, ignoring
ERROR: apport (pid 4857) Sun Dec 16 17:04:33 2018: called for pid 4856, signal 11, core limit 0, dump mode 1
ERROR: apport (pid 4857) Sun Dec 16 17:04:33 2018: executable: /home/himanshu/Minor/a.out (command line "./a.out")
ERROR: apport (pid 4857) Sun Dec 16 17:04:33 2018: executable does not belong to a package, ignoring
ERROR: apport (pid 5840) Sun Dec 16 19:49:22 2018: called for pid 5839, signal 11, core limit 0, dump mode 1
ERROR: apport (pid 5840) Sun Dec 16 19:49:22 2018: executable: /home/himanshu/Minor/a.out (command line "./a.out")
ERROR: apport (pid 5840) Sun Dec 16 19:49:22 2018: executable does not belong to a package, ignoring
ERROR: apport (pid 1856) Sun Dec 16 23:00:50 2018: called for pid 1855, signal 11, core limit 0, dump mode 1
ERROR: apport (pid 1856) Sun Dec 16 23:00:50 2018: executable: /home/himanshu/Minor/a.out (command line "./a.out")
ERROR: apport (pid 1856) Sun Dec 16 23:00:50 2018: executable does not belong to a package, ignoring
ERROR: apport (pid 1874) Sun Dec 16 23:02:37 2018: called for pid 1873, signal 11, core limit 0, dump mode 1
ERROR: apport (pid 1874) Sun Dec 16 23:02:37 2018: executable: /home/himanshu/Minor/a.out (command line "./a.out")
ERROR: apport (pid 1874) Sun Dec 16 23:02:37 2018: executable does not belong to a package, ignoring
root@himanshu-VirtualBox:~/Minor#
```

Fig. 9.1.7: Selecting the log records option and displaying the contents of the accessed log record.

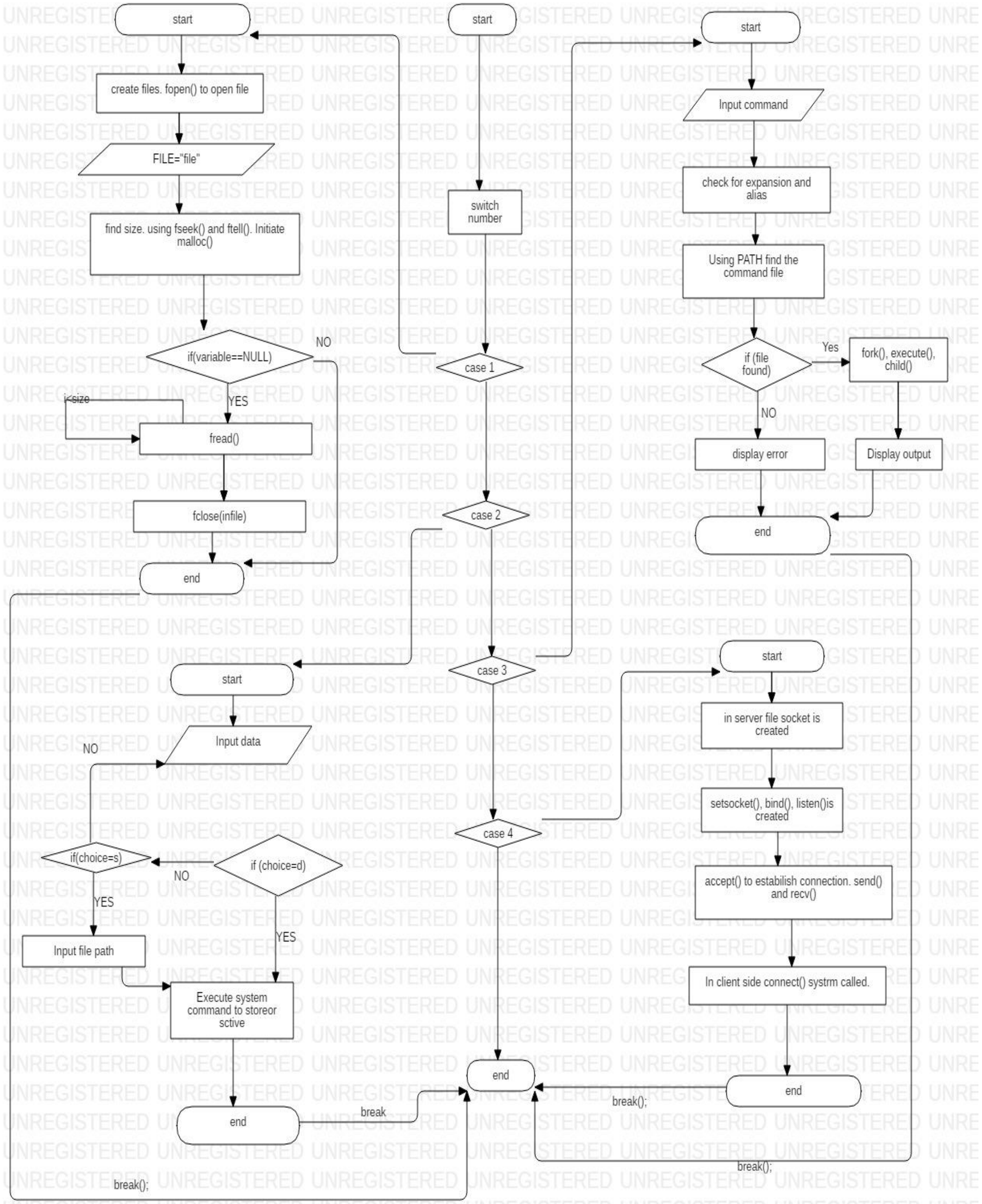
10 Schedule (PERT Chart)


Problem statement
Literature survey
Resource and information gathering
Study about the basic working of the OS, its utilities, functioning of the system log
Mid-August 2018 to first week of September 2018
Dividing project into module
Device and activity tracer module.
Log Monitoring Module
First week of September 2018 to Third week of October 2018.
Testing and implementation of algorithms
Integration of modules
Fourth week October 2018 to First week of November 2018.
Testing and validation
Report completes.
Second week of November 2018 to end week of November 2018.
Documentation

11 Use case Diagram


Activity Fetcher
User Interface
Developer
User of the Modules
Running Time selected by pressing u
Log Records selcted by pressing o
+input choice
Choose from Available options
File list selected by pressing f
process list selected by pressing p
timespot selected by pressing c

12 Flowchart


start
create files. fopen() to open file
FILE="file"
find size. using fseek() and ftell(). Initiate malloc()
if(variable==NULL)
NO
YES
i<size
fread()
fclose(infile)
end
start
switch number
case 1
case 2
case 3
case 4
end
start
Input command
check for expansion and alias
Using PATH find the command file
if (file found)
Yes
fork(), execute(), child()
NO
display error
Display output
end
start
Input data
NO
if(choice=s)
NO
if (choice=d)
YES
YES
Input file path
Execute system command to storeor sctive
end
break
break();
start
in server file socket is created
setsocket(), bind(), listen()is created
accept() to estabilish connection. send() and recv()
In client side connect() systrm called.
end
break();
break();

13 Results and Discussion

Security is the very much debated topic in the recent times globally as it poses a threat to the user's data, unauthorised access to the data and many other issues. If we are unable to find which program/file was when created, modified or deleted than the problems related to these files will be very difficult and unmanageable. In recent times with the advent in the technology and the access of the computers to the masses the issue of privacy is the priority of everyone.

In the project we have analysed the various strategies that can be used to monitor and track the user's activity. Some of the techniques include video recording of the sessions, log collection and analysis, network packet inspection, keystroke logging, kernel monitoring and File/Screenshot capturing etc. We have mainly focused only on the two aspects in our project that are accessing of the log files and creating of the commands that will help to trace the user activities. Rest of the ways are equally applicable and depends on the user requirements, need, and the available resources.

So basically, the results of our project can be summarized as below:

- Access of the log files will give us the details when an application/file/program is created, modified or deleted etc. This will help us to find us the root of the problem that may occur in the file. Thus, we can trace back the steps and find out the cause of the problem and take the required steps. Sometimes we can simply undo the changes done to resolve the problem with the help of the log files.

- Various command lines that will give user the power to implement various functionalities to track the record of the user activities and on the basis of these take precautionary steps.

14 Conclusion

At the end of the project we would like to conclude by saying that the user security, privacy and the protection of user data is a major issue existing in the world and need to be looked after soon before its too late. Our project is a model which address this issue in a comprehensive way. It would prove very beneficial in the real world as the project can work in dynamic conditions. It provides a very optimized solution to this problem very focused on the goal of monitoring the system and user activities.

There are various techniques came to our knowledge regarding the monitoring of the system. We can use our approach in integration with various others approaches discussed. It will finally come to the users need. And according to the users need our developed approaches can be modified.

Through this project we developed the understanding of the working and the basics of the Linux operating system. We analysed the need of monitoring the system and solve the issues related to security, unauthorised access and other threats. In our project we tried to address these issues. Hence, we achieved the objectives proposed above at the end of the project.

15 Future Scope and Strategy

This application is basically a Activity tracer based on Linux operating system which is used to trace the activity done by the user or administrator. This application can be used as a defence mechanism against data breaches and other cybersecurity compromises. Many IT security teams lack visibility into how their users are accessing and utilizing sensitive data, leaving them susceptible to insider threats or outside attackers who have gained access to systems. The further concept that can be added can be:

- Linux operating system allows users to create commands and execute them over the command line. To create a command in Linux, the first step is to create a bash script for the command. The second step is to make the command executable.
- Socket programming for the networking commands.
- Shell Programming for creating own shell for adding functionality of standard UNIX and shell commands.
- We can implement validation part at every point of accessing file for making it more secure so that it can be used by company as a safety for their employees.
- We can integrate it with Big Data, so that the application will learn the pattern of the user and any ambiguous pattern will block the access of the file.
- We can integrate with an Android application to get notified if there are any changes onto the system.

16 References

[1] https://www.geeksforgeeks.org/basics-file-handling-c//.

[2] https://www.howtoforge.com/tutorial/ubuntu-performance-monitoring//.

[3] https://www.tecmint.com/command-line-tools-to-monitor-linux-performance//.

[4] https://blog.serverdensity.com/80-linux-monitoring-tools-know//.

[5] https://en.wikipedia.org/wiki/Log_file//.

[6] https://www.solarwinds.com/topics/user-activity-monitoring//.

[7] https://digitalguardian.com/blog/what-user-activity-monitoring-how-it-works-benefits-best-practices-and-more//.

[8] Web site activity monitoring system with tracking by categories and terms.
Author: Janet Yoo, Kian-Tat Lim, Stanley Ben Wong, Elliott Yasnokvsky.
Current Assignee: ENERGETIC POWER INVESTMENT Ltd.
Publication Number: US7146416B1.

[9] Methods and systems for monitoring user, application or device activity.
Author: Eric Anderholm, David Losen
Publication Number: US20050183143A1
Current Assignee: SERGEANT LABORATORIES Inc

[10] https://en.wikipedia.org/wiki/User_activity_monitoring#Capturing_activity//.

[11]https://searchsecurity.techtarget.com/tip/How-to-know-if-you-need-file-activity-monitoring-to-track-file-access//.

[12] https://en.wikipedia.org/wiki/Computer_and_network_surveillance//.

[13] Computer-user activity tracking system and method. Author: Gene Goykhman Current Assignee: Gene Goykhman Publication number: US20020174134A1

[14]https://www.creditcards.com/credit-card-news/credit-card-security-id-theft-fraud-statistics- 1276.php//.