OS Project Report
Processes and Memory Management Terminal

M K Laksath Adityan (B19CSE045)
Himanchal Sharma (B19CSE039)
Harsh Kumar Meena (B19CSE037)

Introduction :

As we all may know that Linux operating systems are vast and complicated.The number of commands that are there for a particular functionality is not just one but many with minor changes.We propose a method that helps visualise the storage facility of the linux operating systems in a better systematic way.

Objective :

1.The problem of displaying the total disk space and its components is to be addressed.
2.The unnecessary files that havent been used in a long time have to be shown in a way to make the user decide whether to keep them or not.
3.All the current processes that run in a linux system are displayed in linux task manager.It does not address the problem of closing all the processes for a particular application.It does not list down those processes that have to be terminated to completely ensure that the program does not run anymore.

Methodology :

We have addressed these issues with our own custom made command prompt which not only performs the inbuilt commands but also looks after these issues raised with our own commands.

We first created the command prompt using the help of pipes and virtual files.
When a user enters a command we execute it in our code and store them by overriding the exec() command.Inorder to denote that this is a custom made terminal running, we have given our default colorings with a black background for the text.

The heading was aligned at center using the ioctl(STDOUT_FILENO, TIOCGWINSZ, &w); command which returns the width of the command prompt.

Arrows that cause ^[[A] sometimes in terminal are also prevented.We have used getch() in our terminal to input each and every character.The arrows are analyzed and are removed accordingly.

We address these problems into three major components :

1.pcinfo :

- Lists down the storage status of the operating system (In a more organised way that is easier to understand).
- We get the information from /proc/cpuinfo folder and write it down at couinfo_.txt.
- We also use the df command and write down the results in df.txt.
- Cpu and df are combined using the python files cpu.py and df.py respectively.
- From the large texts, we extract processors,model_name,total_cores,total_caches, and cpu_MHz in cpu.py from the file cpuinfo_.txt.
- In df.py, we extracted information from df.txt to calculate the free space in bytes, Gb and percent availability and it is displayed.

2.findall:

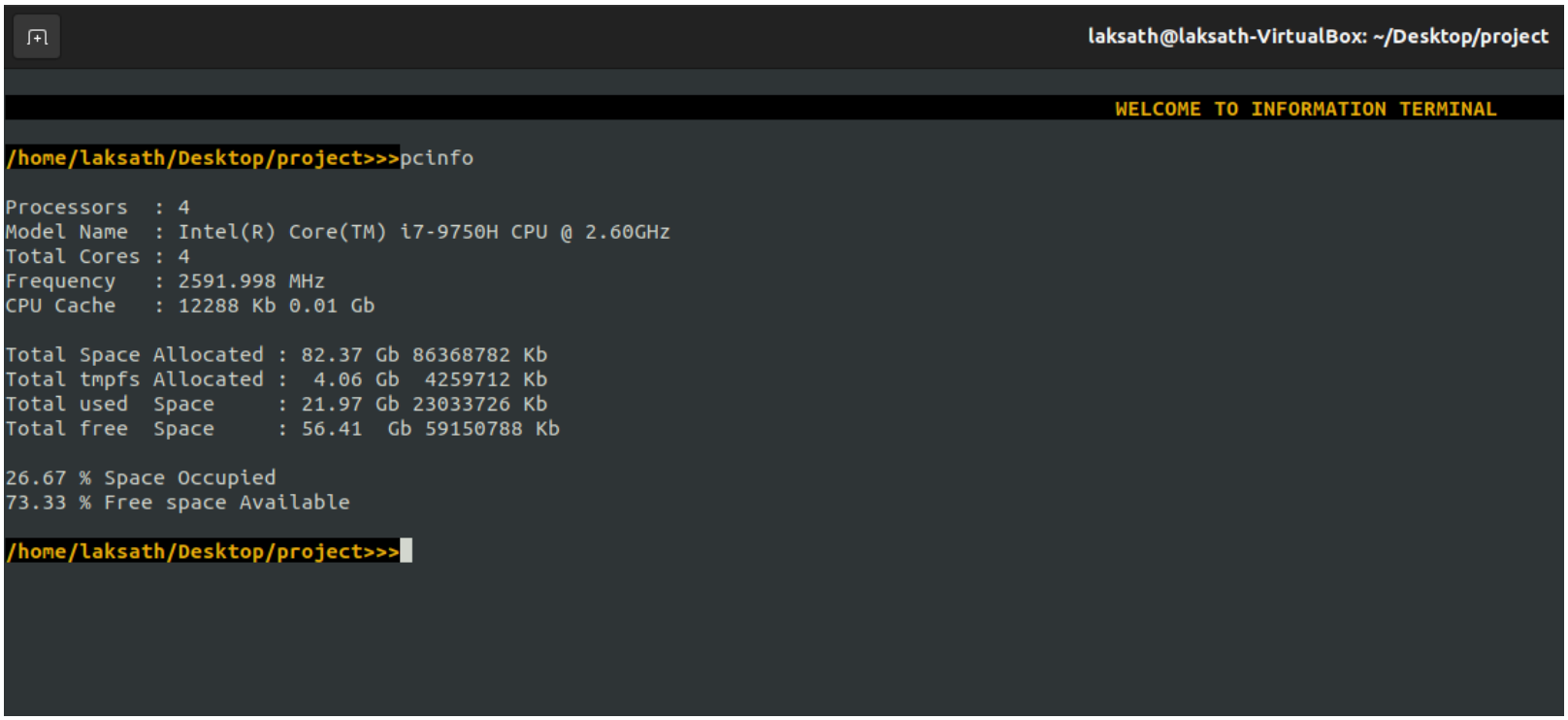
- This command works on any directory which when applied, produces a bar graph that shows the space occupied by each directory/file present in that directory.
- It also shows the last time we have opened a particular file/folder (this way, it helps us know if that file would be unnecessary).
- We have used the plotex library in python to plot the graph and the last used was found using the ls command and only those lines were extracted for a better visualisation.

3.manager:

- Shows the status of all currently running processes.
- We have also provided a custom made functionality to terminate them.
- If we give the name of the application, it would show all its corresponding processes making it easier for us to decide whether to keep/kill that process.
- Advance Option is also provided which constantly updates the terminal with the running processes.
- Firstly , the table is obtained through the top command which is then written into mem_stat.txt file.
- The alignments that are incorrect from the inbuilt command.Hence, they are rectified and are adjusted and changes are made accordingly to display a neat table with process ID and the application’s name.
- Then , if we decide to kill an application, we firstly need to terminate all the processes that are related to that particular application.
- Therefore, we input the user’s choice of application to be terminated.
- From the mem_stat.txt file, we go line by line and look out for all the processes that belong to that application and we return the processes that belong to that application along with its process IDs.Exisiting linux systems do not have this functionality and Hence we implemented it.
- Then we used kill -9 in our code to terminate a process ID given its process.

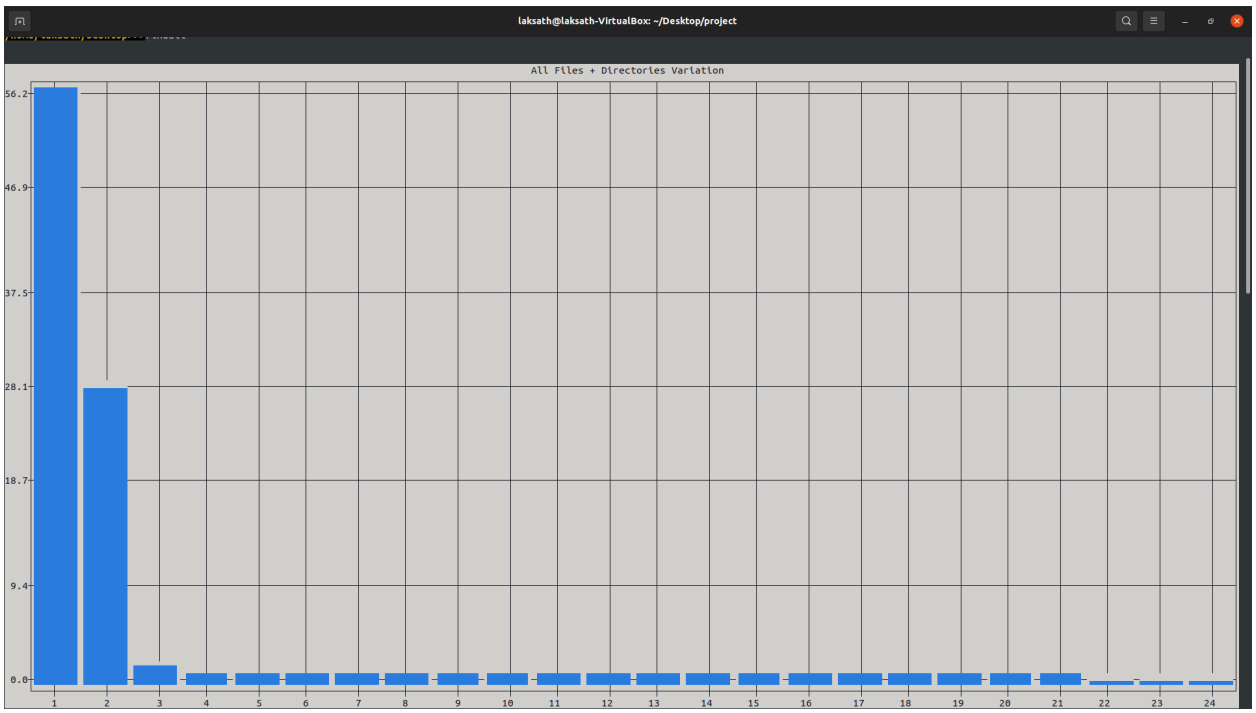
Results :

(i) For pcinfo, regardless of the directory one is in, the following details are displayed :



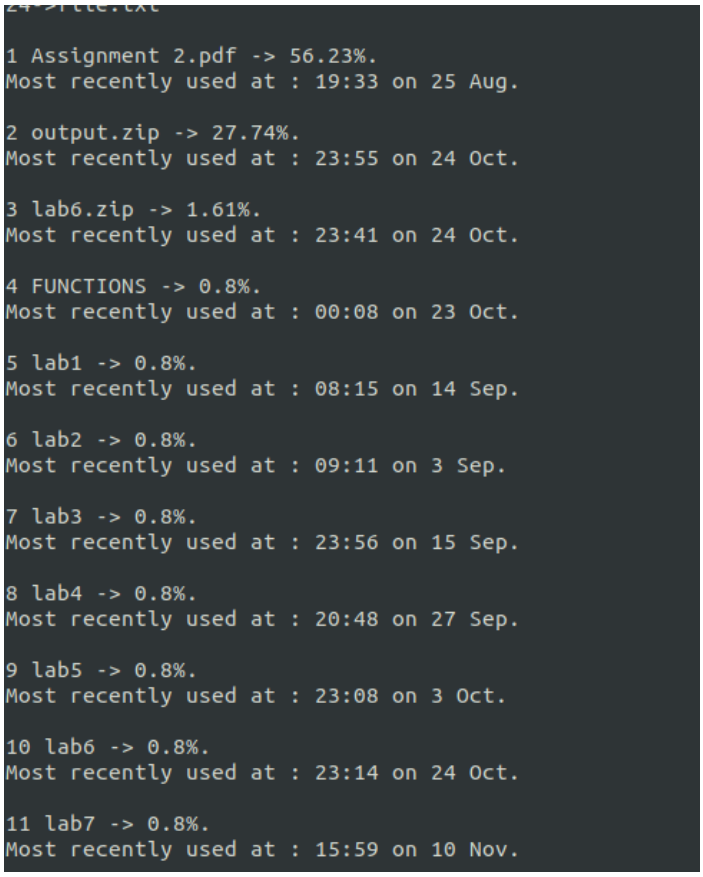
The picture is self-explanatory of the functionalities.

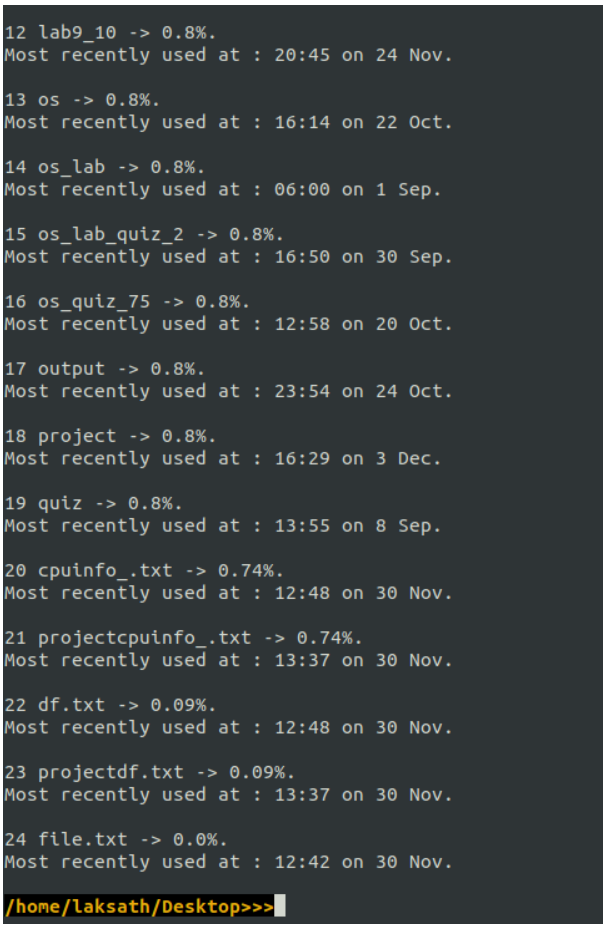
(ii) For findall on Desktop directory, we notice the following results :



Firstly, we observe a bar graph plot.
The y axis is the percentage of space occupied of a file/directory.
The x-axis is a range of numbers starting from 1 to n where n is the total number of files/folders in that directory.

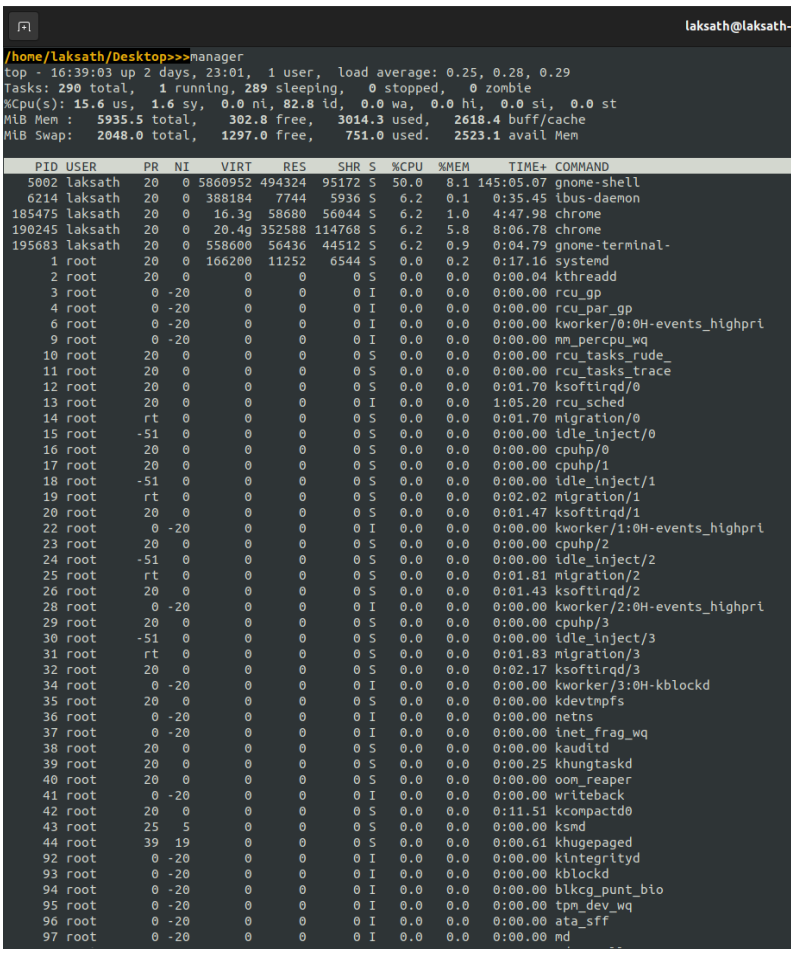
Later on , the file/folder corresponding to that particular index in the x axis is displayed.

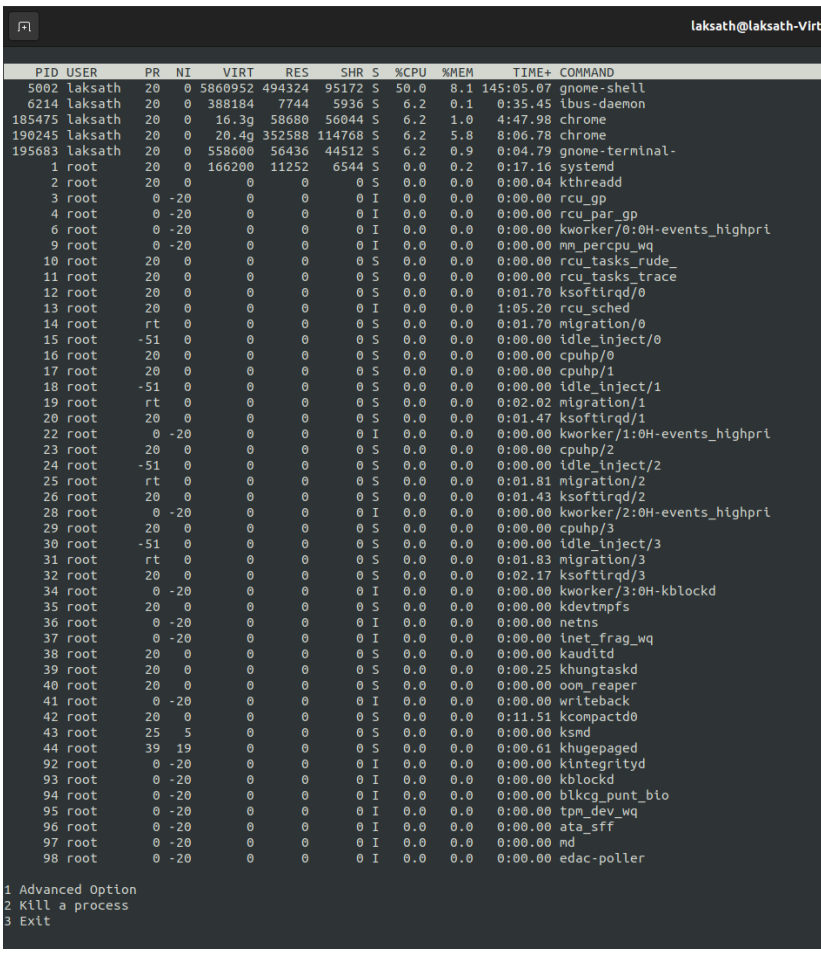




We then displayed the Most recently used at to with the time and the date along with the month.This way, it helps us know if that file would be unnecessary.

(iii) For manager , we notice the following :



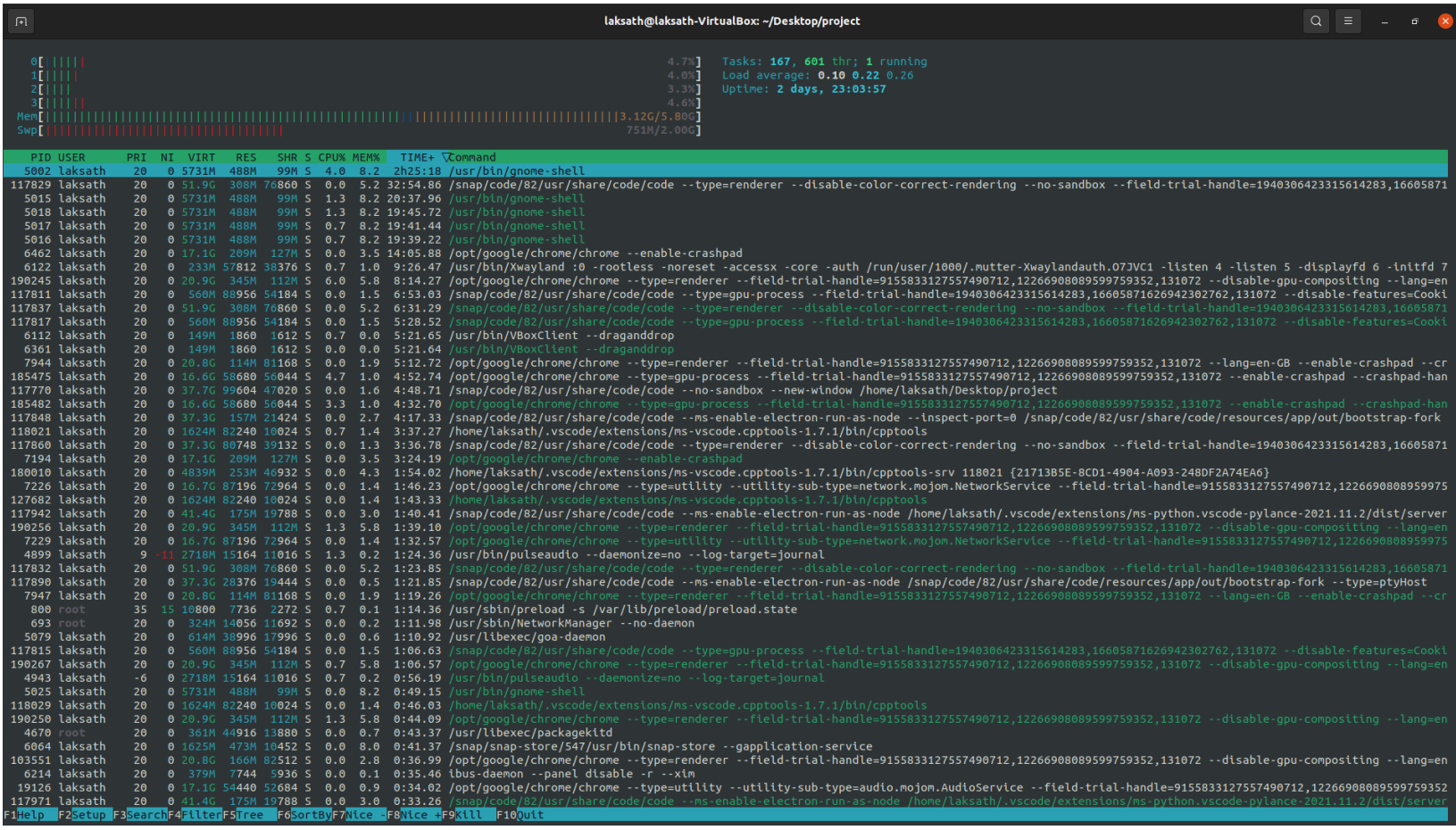


On entering manager, we create the following.
It displays all the processes along with the process id and the name of the application the process belongs to.

As you may notice,
The third and fourth line both belong to the chrome application.
Killing either one of the processes does not guarantee the process termination.
Therefore , we need to kill all the processes that belong to chrome.

1 Advanced Option :

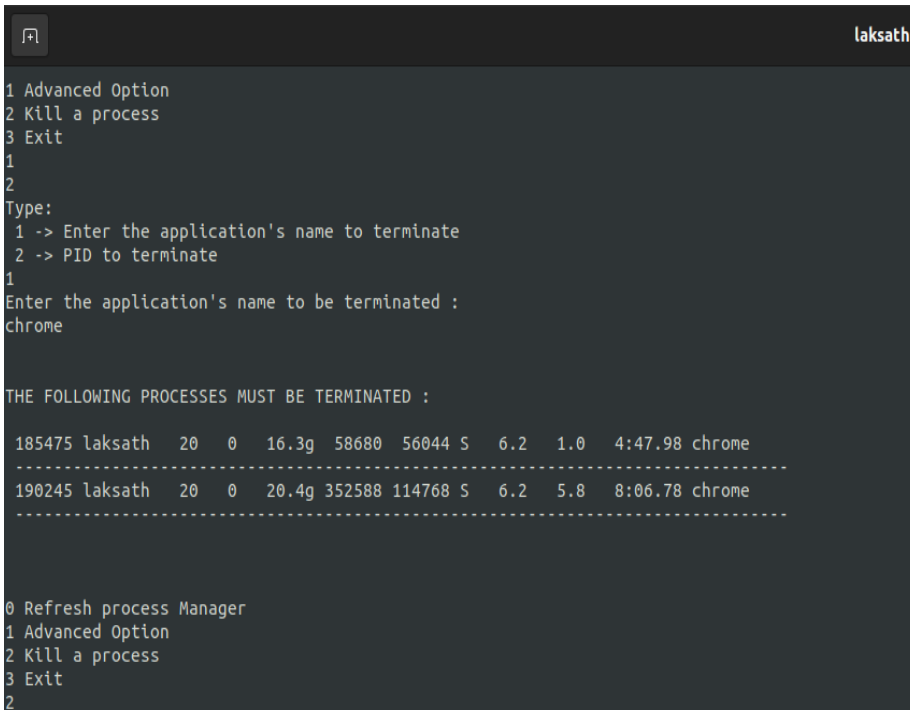
Choosing the Advanced option provides the info about all the processes that are changing constantly with too much information about each and every process that runs in the background.

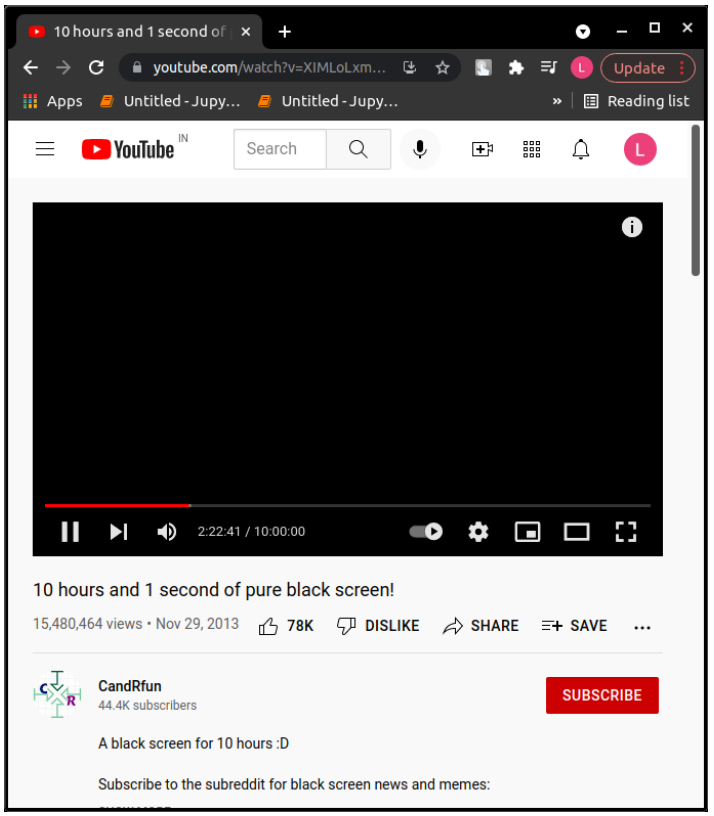


2 Choosing to kill a process provides 2 choices.

We can either directly enter the process ID (2nd option) (or) If we are not sure, we shall choose the first type and enter the application's name to be terminated (chrome in this case) (option 1).

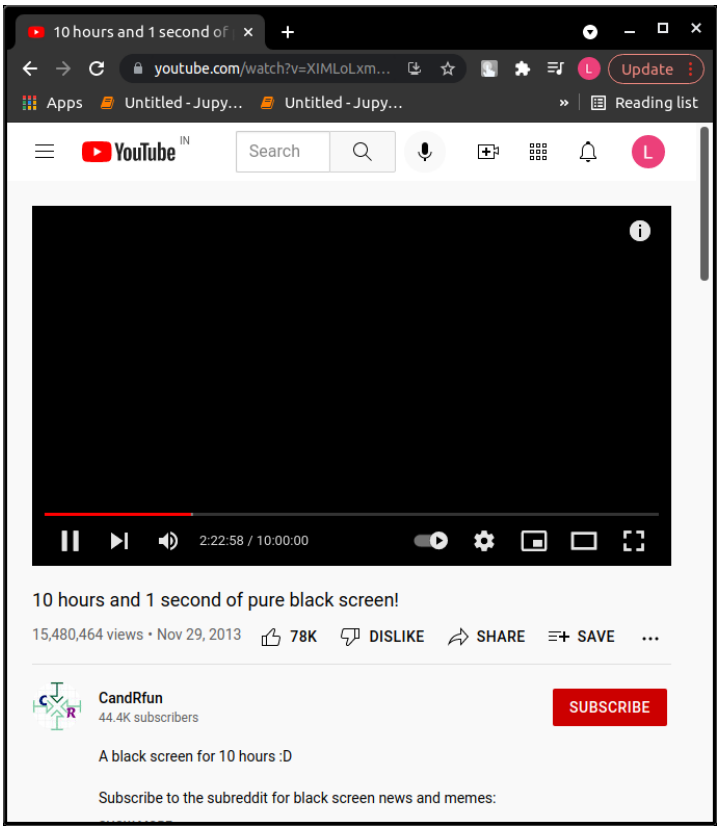
We dont delete all the processes, we rather display the information about it.





Here we have chosen option 1.We wanted to kill all the chrome processes.
Here, as we can see, we have a chrome application with a Youtube video running.
There are 2 processes that are running with pids : 185475 and 190245 respectively.

```
2
Type:
 1 -> Enter the application's name to terminate
 2 -> PID to terminate
2
Enter the process ID to be terminated :
185475
0 Refresh process Manager
1 Advanced Option
2 Kill a process
3 Exit
2
```

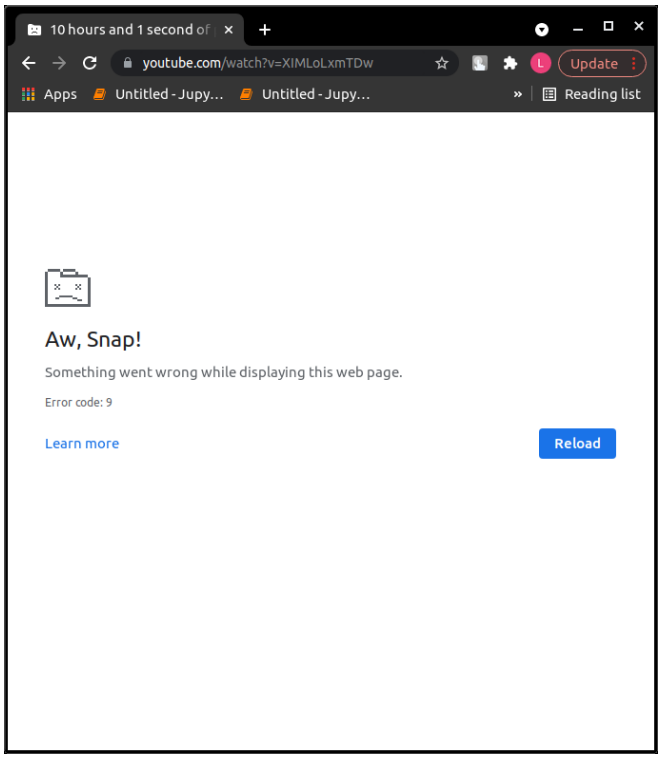


Now to terminate these 2 processes, we need to choose type 2 in the Kill a process option.

We specified process ID 185475 to terminate and we can see the video is still playing which means the chrome process is not yet terminated.
This is because it is important that we also kill the other process which makes sure that the entire chrome process is shutdown.

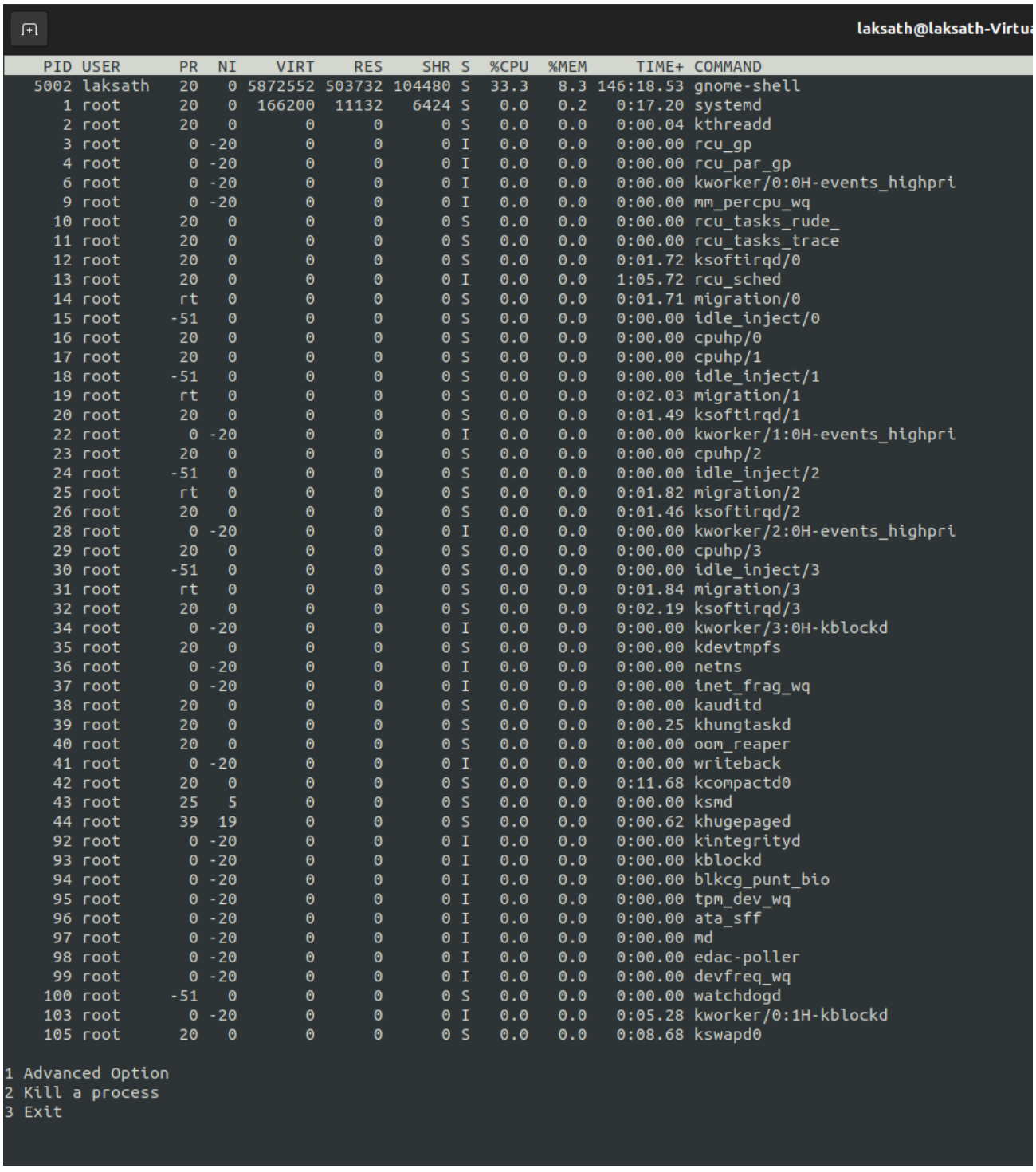
```
Type:
 1 -> Enter the application's name to terminate
 2 -> PID to terminate
2
Enter the process ID to be terminated :
190245
0 Refresh process Manager
1 Advanced Option
2 Kill a process
3 Exit
0
top - 16:39:03 up 2 days, 23:01,  1 user,  load average: 0.25, 0.28, 0.29
Tasks: 290 total,   1 running, 289 sleeping,   0 stopped,   0 zombie
%Cpu(s): 15.6 us,  1.6 sy,  0.0 ni, 82.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   5935.5 total,    302.8 free,   3014.3 used,   2618.4 buff/cache
MiB Swap:   2048.0 total,   1297.0 free,    751.0 used.   2523.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   5002 laksath   20   0 5860952 494324  95172 S  50.0   8.1 145:05.07 gnome-shell
```



- Now, after we terminated the chrome process with process ID 190245, we can see that the video stopped playing as that process has been killed by our program.
- Error Code 9 appears because the process has been killed suddenly from our local operating system.
- This is how we completely terminate applications running in linux systems.
- However, the existing softwares fails to provide these functionalities and therefore, we have shown the above implementations.

- Once again , to check if the chrome processes are still running, we can refresh the Process Manager by typing in 0.



As we can notice, the chrome process is not there anymore.

Link to the project :
https://drive.google.com/drive/folders/1lVWGa0-lAV7aFNx4cIXWEJZAAZlILx3v?usp=sharing

How to run :

1. Go to the project's directory.
2. run : g++ cmd.cpp; ./a.out;

All the other python files are linked to it directly.
So, this single command is sufficient to run the entire project.