NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

(KARACHI CAMPUS)
FAST School of Computing
Spring 2024

PROJECT REPORT

TITLE: TASK MANAGER

GROUP MEMBER NAMES:

SYEDA FAKHIRA (22K-4413)

AAFREEN (22K-4448)

INSTRUCTOR: MS ATIYA JOKHIYO

TABLE OF CONTENTS

- INTRODUCTION (AIM OR MOTIVATION) 3

- BACKGROUND (RESEARCH & PROJECT SELECTION) 3

- PROJECT SPECIFICATION 3

- PROBLEM ANALYSIS 3

- SOLUTION DESIGN 4

- IMPLEMENTATION & TESTING 7

- PROJECT BREAKDOWN STRUCTURE 8

- RESULTS (OUTPUTS SCREENSHOTS) 8

- CONCLUSION (SUMMARY & DISCUSSION) 14

➢ INTRODUCTION (AIM OR MOTIVATION)

Task Manager provides information about hardware resource usage and performance as it relates to the system's individual apps and processes, including services. The information includes usage details about the system's CPU, memory, disk, network and graphics processing unit (GPU) resources. An Operating System is in charge of a task, which is the basic building block of programming. It acts as a monitoring tool to assist users in understanding how resources on their system are used by applications, processes, and services.

➢ BACKGROUND (RESEARCH & PROJECT SELECTION)

The project aims to develop a system monitor tool using GTK and C programming language. The tool provides various functionalities such as monitoring CPU, memory, disk, network, and GPU usage, managing processes and services, displaying system information, and implementing scheduling algorithms.

➢ PROJECT SPECIFICATION

Using the GTK library, a graphical user interface (GUI) application is created for the project to:

1. Display information regarding cpu, disk , gpu, memory usage.
2. Display system information and details of all the processes.
3. Control processes [start, suspend and kill processes].
4. Service Management.
5. Use scheduling with round robin to schedule processes.
6. Arrival, burst, and completion time of processes are also calculated.

The implementation of system calls are done by various libraries. CLI(Command line interface) is used to run the code. Threads are also used in the code to call a function when a button is pressed.

➢ PROBLEM ANALYSIS

The project is created as an extensive monitoring tool that gives users' access to real-time information about how system resources are being used. It enables users’ to effectively manage services and processes.

➢ SOLUTION DESIGN (PROJECT DETAIL, FUNCTIONALITY AND FEATURES)

The system monitor tool is designed to have a user-friendly GUI interface using GTK. It includes functionalities like:

○ LIBRARIES USED:

- ➢ #include <gtk/gtk.h>
- ➢ #include <stdio.h>
- ➢ #include <stdlib.h>
- ➢ #include <unistd.h>
- ➢ #include<pthread.h> //for thread creation
- ➢ #include<errno.h>
- ➢ #include<signal.h>
- ➢ #include<sys/types.h>
- ➢ #include<string.h>
- ➢ #include<time.h>

○ HEADER FILES CREATED:

- ➢ #include "cpu_utilization.h"
- ➢ #include "SYSTEM_INFORMATION.h"
- ➢ #include "MEMORY_USAGE.h"
- ➢ #include "SERVICE_MANAGEMENT.h"
- ➢ #include "PROCESS_DETAILS.h"
- ➢ #include "NETWORK_USAGE.h"
- ➢ #include "GPU_USAGE.h"
- ➢ #include "DISK_USAGE.h"
- ➢ #include"process.h"

○ FUNCTIONS USED AND ITS DETAIL:

- ➢ Void killProcess(): to kill a specified process
- ➢ Void stopProcess(): to suspend a specified process
- ➢ Void ResumeProcess(): to resume a process
- ➢ Void *storeCurrentProcess(void *arg): store all running processes into an array of process
- ➢ Void *calcABCTime(void *arg): to calculate arrival,burst, and completion time
- ➢ Void *schedullig_algo(void *arg): Process scheduling algorithm is used for running processes

- LIBRARIES FUNCTIONS USED:

➢ Void *network_usage(void *arg):

Command -> "nload -m" is run to show multiple devices at a time. Command -> "nload -t" is run. It determines the refresh interval of the display in milliseconds. If the system call fails to execute, an error is displayed.

➢ Void *gpu_usage(void *arg)

Command -> "sudo lshw -C display" is used to

➢ Void *cpu_utilization(void *arg)

Information regarding CPU usage is extracted from the `/proc/stat` file. System call "sudo lshw -C display" is run and output is displayed on the console, system_monitor file and gpu_usage file.
User : Time spent with normal processing in user mode.

Nice: Time spent with nice processes in user mode.

Idle: Time spent in vacations twiddling thumbs.

system:Time spent running in kernel mode.

The following formula is used to calculate the cpu usage:

delta_used = (user2 - user1) + (nice2 - nice1) + (system2 - system1);

delta_total = delta_used + (idle2 - idle1);

cpu_usage = (int)((delta_used * 100) / delta_total);

➢ Void *memory_usage(void *arg)

The file "/proc/meminfo" is read and compared. If key equals total memory, free memory , buffer, cache if match found values are set.

➢ Void *disk_usage(void *arg)

Statvfs is used and a structure is created. Total and available disk space is calculated.

➢ Void *own_process (void *arg)

This function allows users to create processes.

- Void *service_management(void *arg)

  Services are displayed. Users are asked to choose a service. 3 options are given : to start a service, get status of the service and stop the service.

- Void *system_information(void *arg)

  <sys/sysinfo.h> library is used. A structure of sysinfo is created and information is stored in files and displayed on the console as well.

○ GUI FUNCTIONS USED AND ITS DETAIL:

- Void on_cpu_button_clicked(): to calculate cpu usage
- Void on_memory_button_clicked(): to calculate memory usage
- Void on_fork_button_clicked(): to create your own processes
- Void on_process_button_clicked(): to know the detail of process
- Void on_disk_button_clicked(): to know disk usage
- Void on_network_button_clicked(): to display network usage
- Void on_gpu_button_clicked(): to display gpu usage
- Void on_service_button_clicked(): to avail services of linux
- Void on_system_button_clicked(): to know system information
- Void on_suspend_process_button_clicked(): to suspend a process
- Void on_resume_process_button_clicked(): to resume a process
- Void on_kill_process_button_clicked(): to kill a process
- Void on_store_current_process_button_clicked(): to store a current process
- Void on_calc_ABC_time_button_clicked(): to calculate arrival, burst and completion time
- Void on_scheduling_algo_button_clicked(): to schedule processes
- Void on_exit_button_clicked(): To exit the program

○ SYSTEM CALLS USED:

- Sudo systemctl start: to initiate a service of os
- Sudo systemctl stop: to stop the service of os
- Sudo systemctl status: to know the status of current running service
- Sudo nload -m: it shows real time information of network usage
- Uptime, loads, totalram, freeram, sharedram, bufferram, total swap, free swap, total high, free high, mem_unit
- sudo lshw -C display: it is used to list detailed information about the display devices (graphics cards) in the system
- Ps aux: to list down all current processes
- Fork(): to create a child
- Top, htop: These are command-line utilities used for monitoring system resource usage and managing running processes

➢ GOALS

The primary goal is to develop a system monitoring tool with a user-friendly interface. The project contains comprehensive functionalities for monitoring and managing system resources.

➢ TOOLS AND TECHNOLOGIES

The project is implemented in C programming language using GTK library for GUI development. It leverages standard C libraries and custom header files for specific functionalities.The project is done on command line. Packages installed are atop, htop, gtk library.

➢ IMPLEMENTATION & TESTING

The code is implemented according to a real time monitoring system. The code is tested on various commands. GUI implementation is done via library name gtk/gtk.h . Various system calls are implemented through built in libraries such as sys/info, sys/types, signal.h, pthread.h. Errors are dealt with by library errno.h. Overall the project is implemented with built in libraries and system calls. The code is compiled using the following command:

```
gcc mainfile.c -o output `pkg-config --cflags --libs gtk+-3.0`

./output
```

PROBLEMS ENCOUNTER

We encountered issues related to system command execution, process management, Network usage, service management, and GUI interaction. Not enough packages were downloaded in our system to run commands such as htop, atop, nload, top etc. We solved this issue by installing the packages.

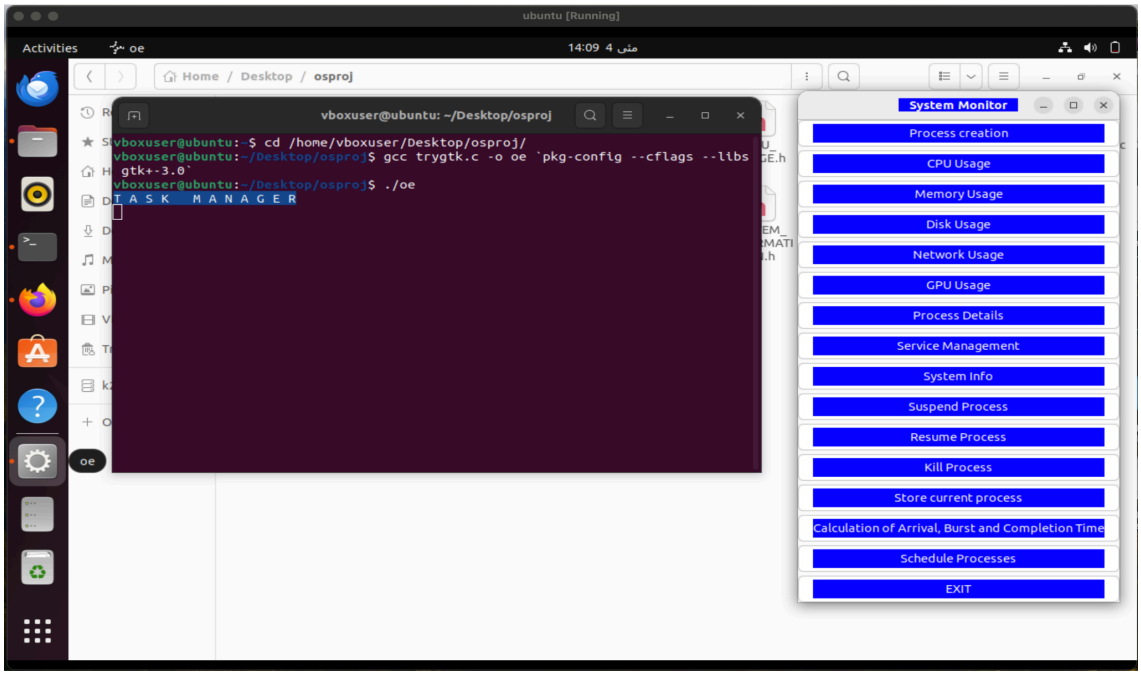
While implementing GUI an issue we encountered was regarding the installation of the gtk library. However we installed it using sudo apt-get install libgtk-3-dev. Another issue we encountered was that we were trying to compile our code with gcc mainfile.c -o output command but it didn’t recognize the gtk library and displayed an error. To fix this issue we did out research and found out that the file was to be compiled using the following command: gcc mainfile.c -o output `pkg-config --cflags --libs gtk+-3.0`

➢ PROJECT BREAKDOWN STRUCTURE (WORKLOAD DISTRIBUTION WITH TIMELINE)

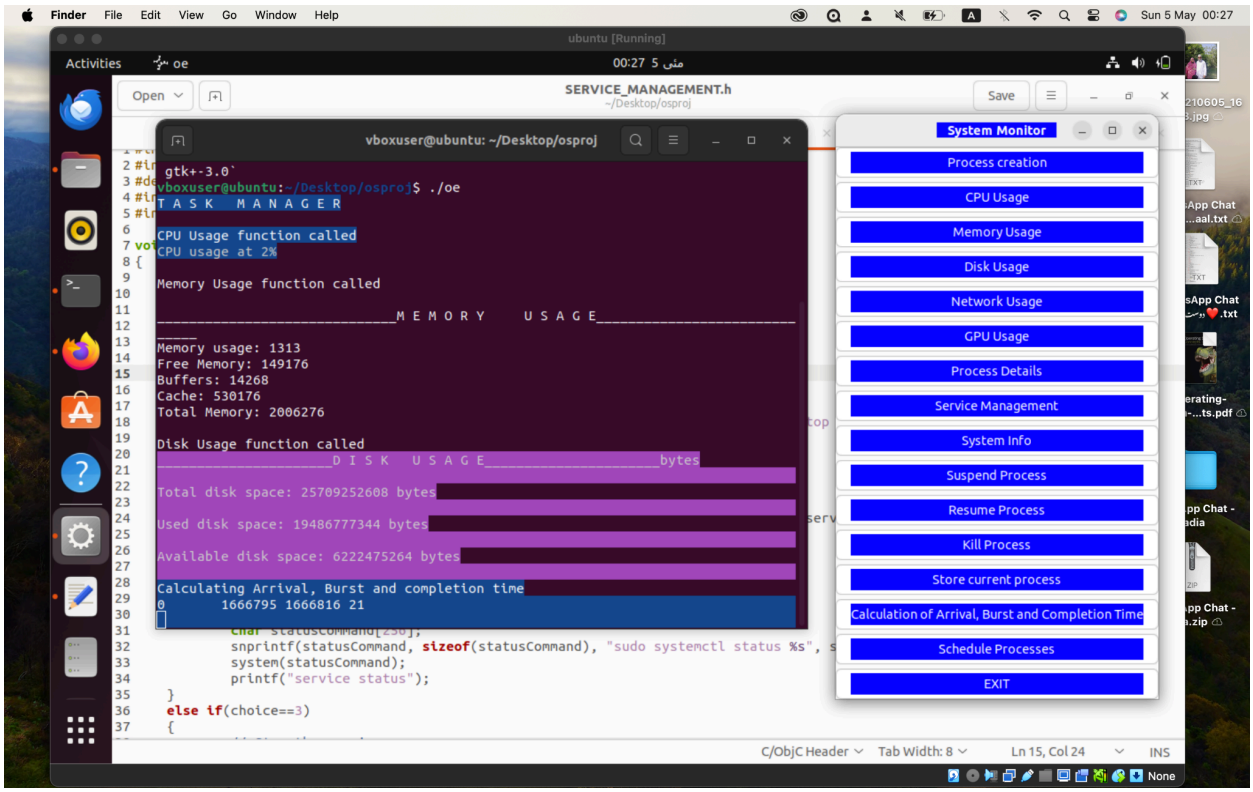
1. Week 1: 15th april: Mutually decided the project structure.
2. Week 2:
    a. Header files made to calculate and display cpu, disk , gpu, memory usage.(Fakhira)
    b. Coding for system information and details of all the processes. (Fakhira)
    c. Functions made to control processes [start, suspend and kill processes]. (Aafreen)
3. Week 3:
    a. Service Management. (Fakhira)
    b. Process scheduling with round robin implemented . (Aafreen)
    c. Function made Arrival, burst, and completion time of processes. (Aafreen)
    d. Gui added to the project .

➢ RESULTS (OUTPUTS SCREENSHOTS)

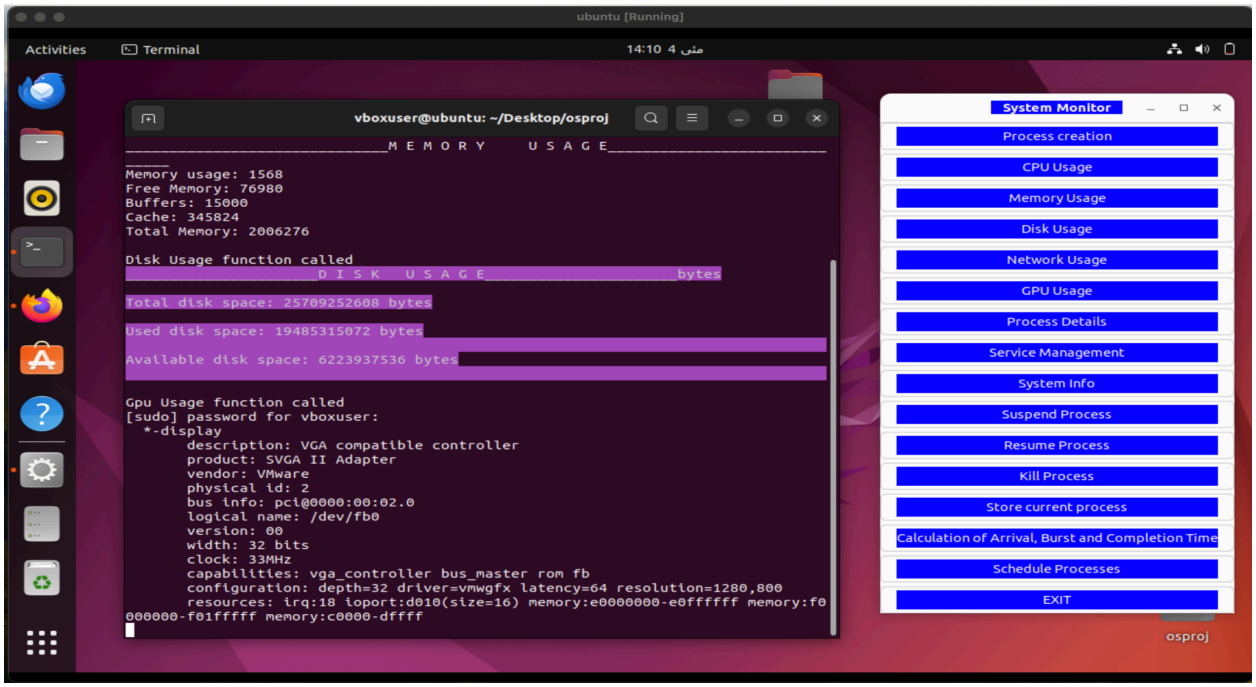
Displaying the main display of the project



Cpu usage function called

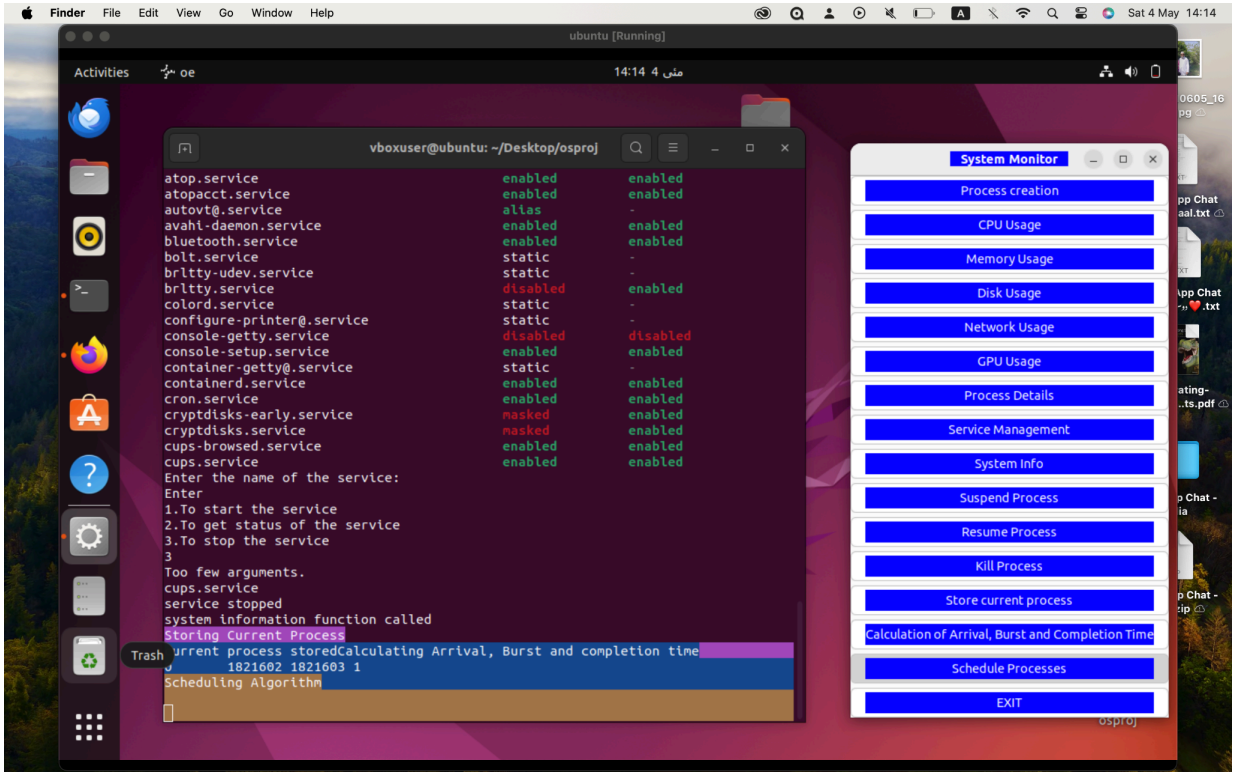


Memory usage ,disk usage and gpu usage function called

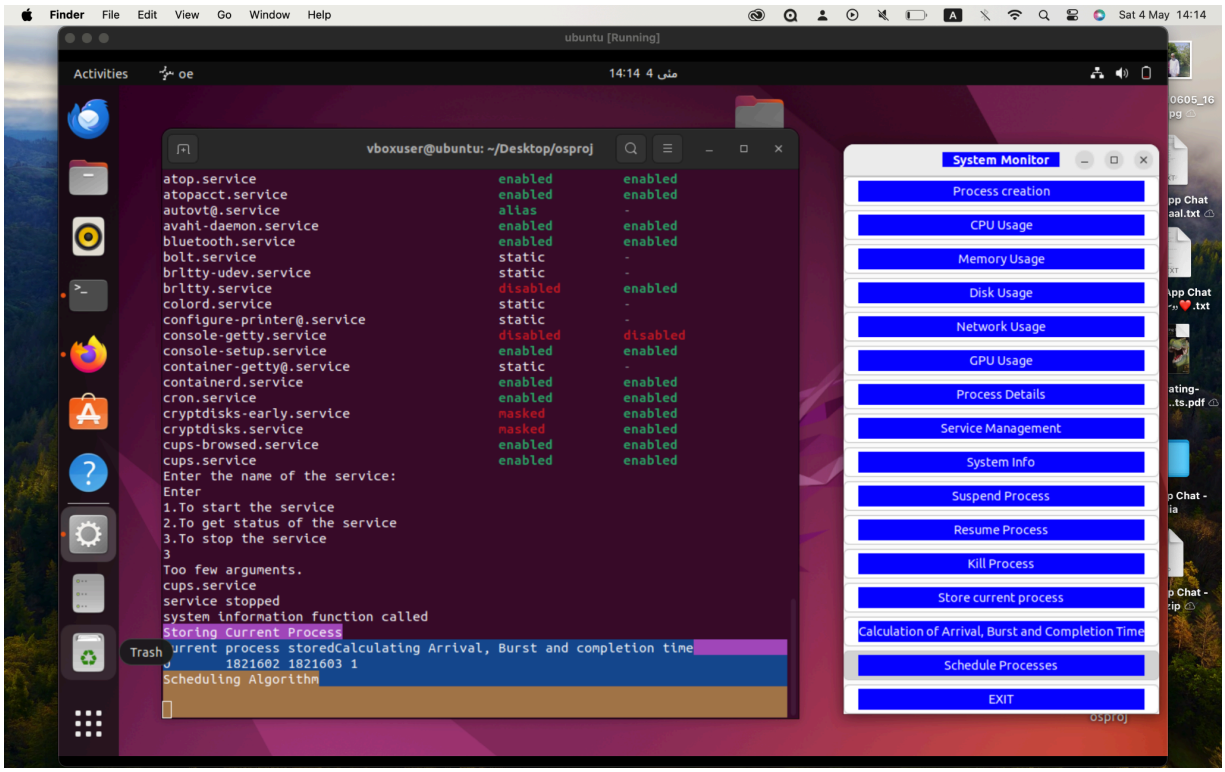


System services management function called

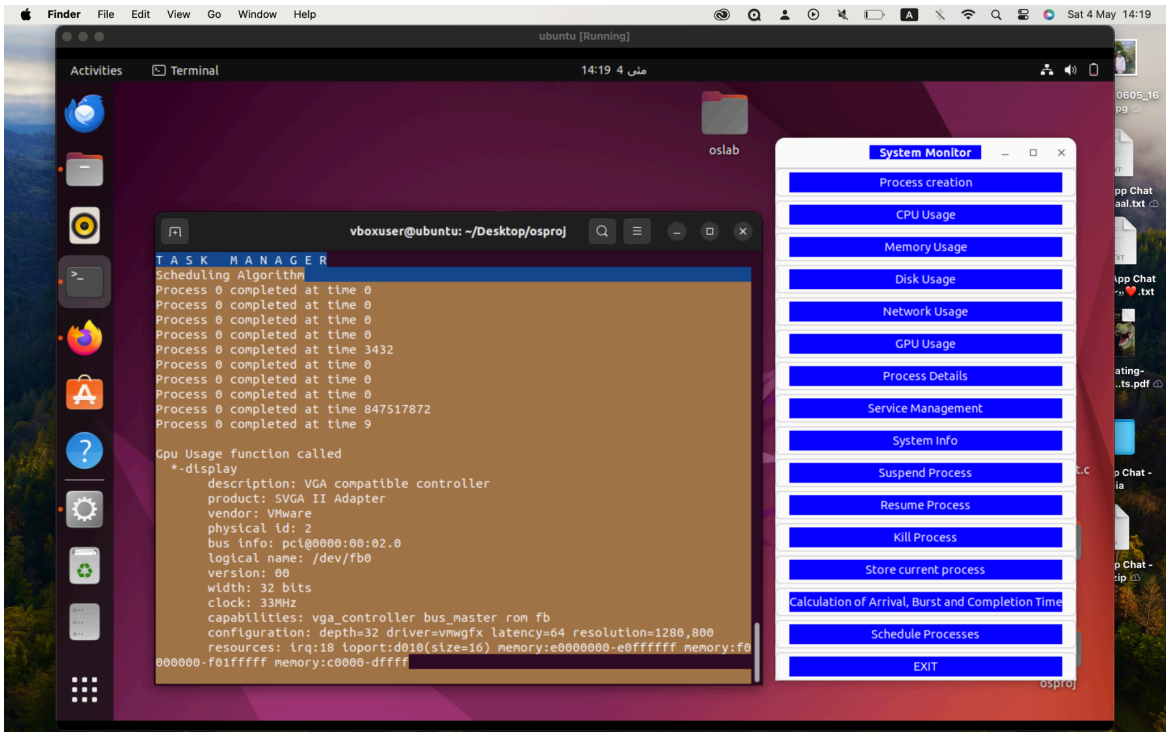
Calculation of arrival burst and completion time



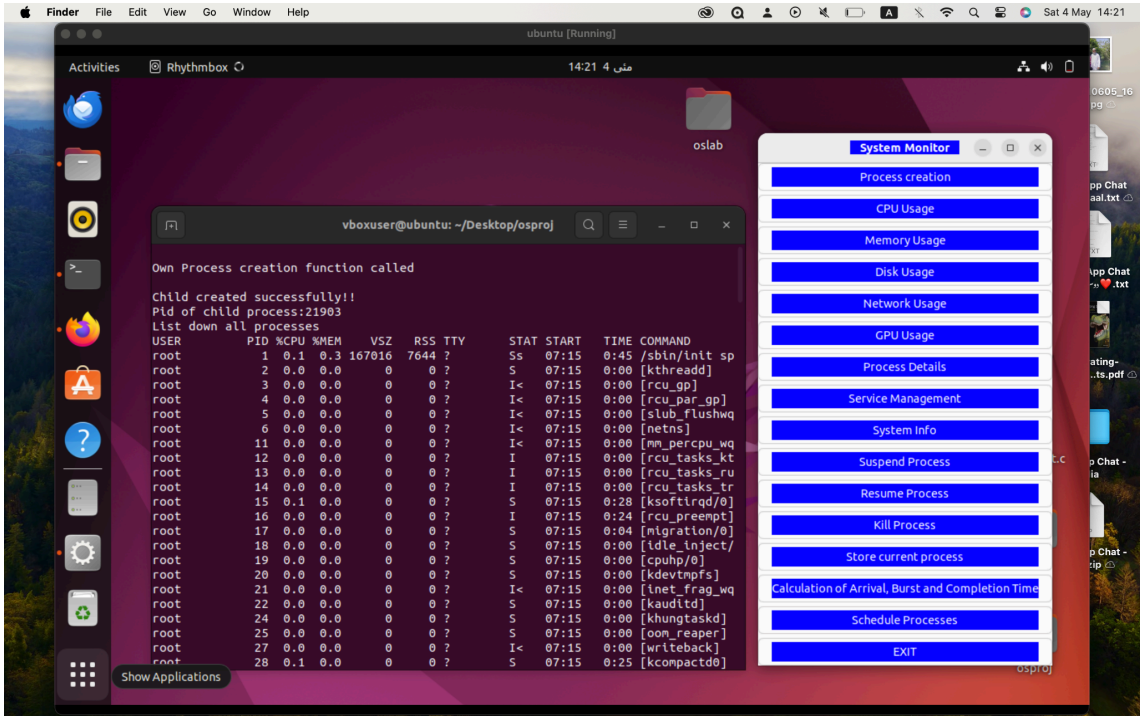
Create your own process and scheduling algorithm function is called



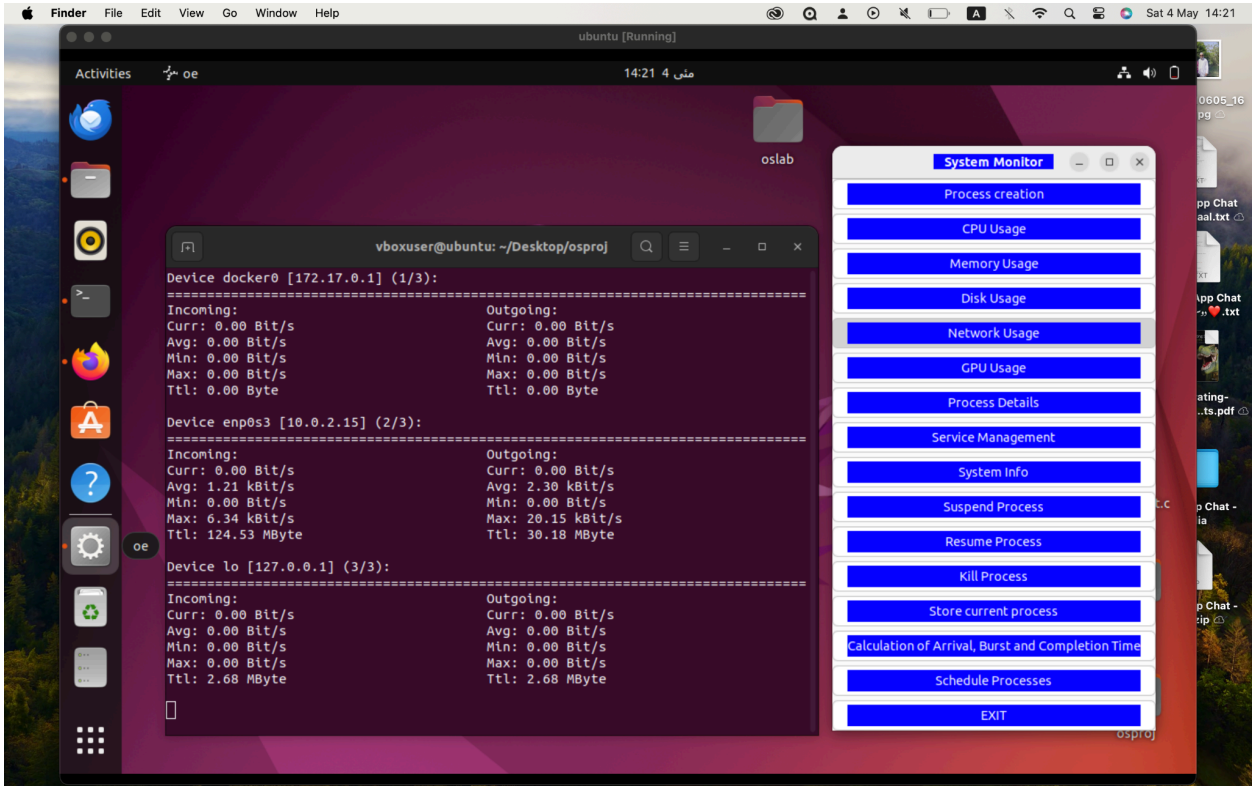
GPU usage function called



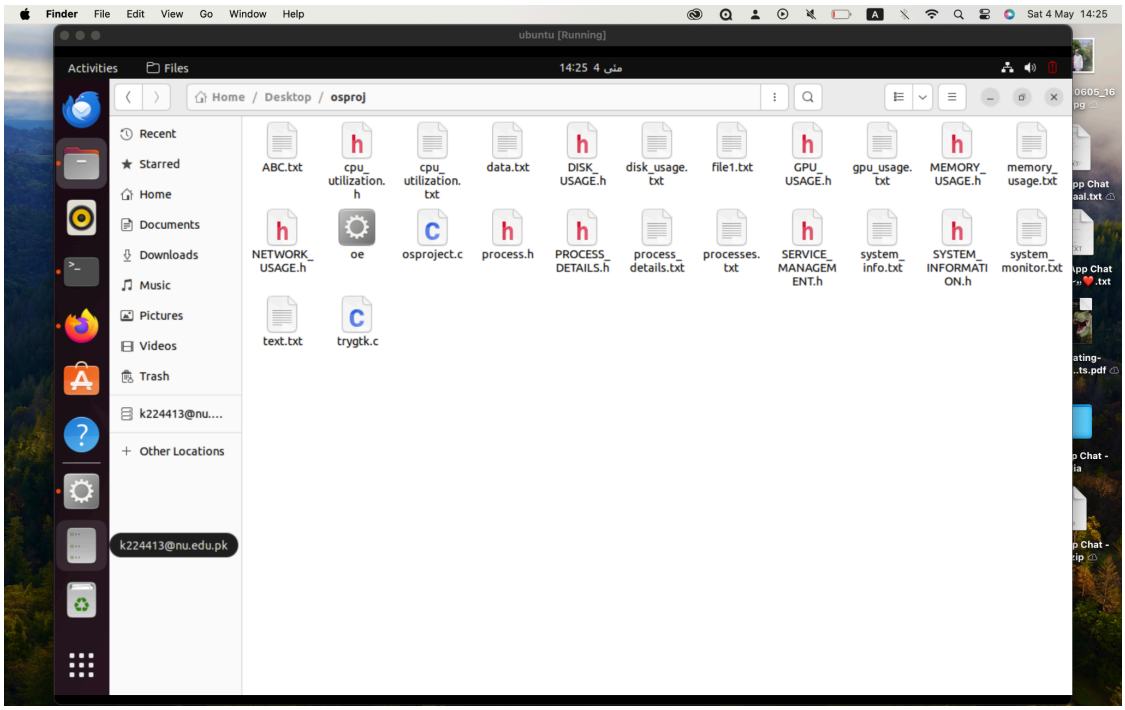
Own process creation function is called



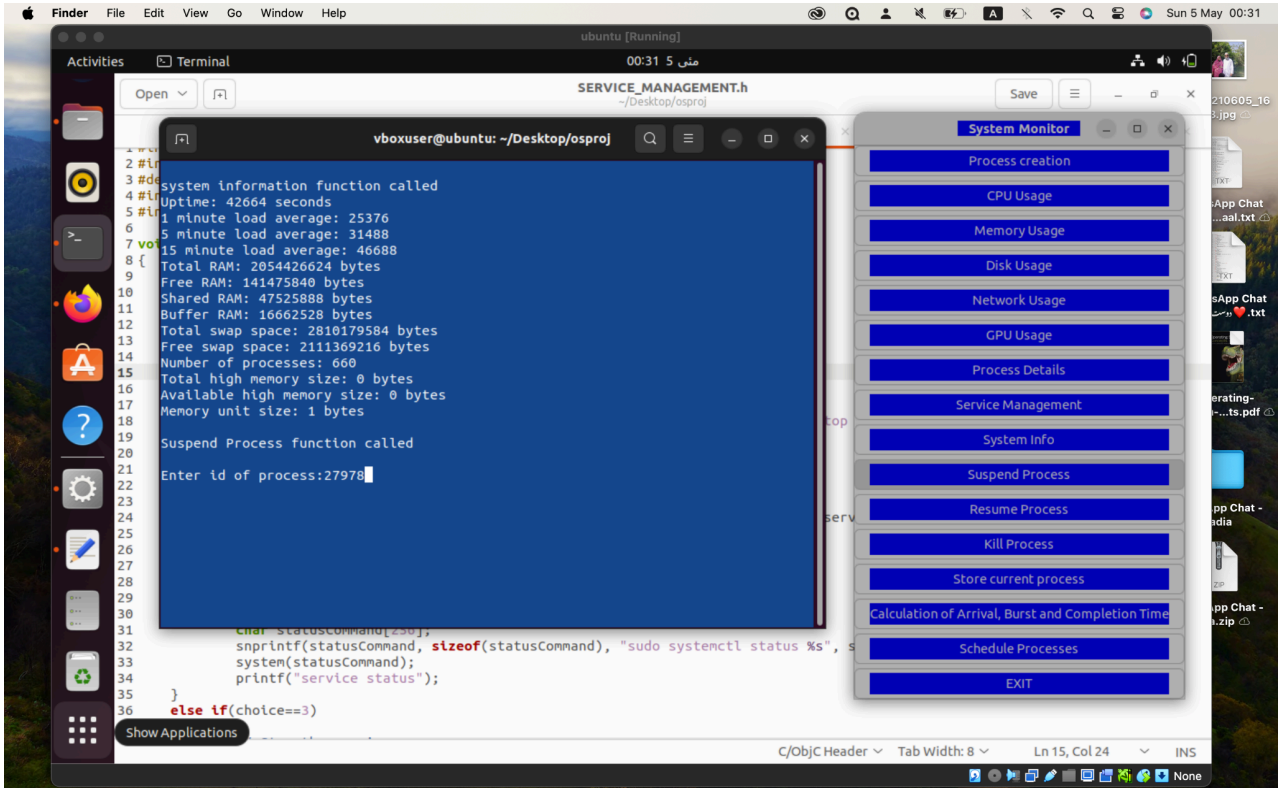
Network usage function is called



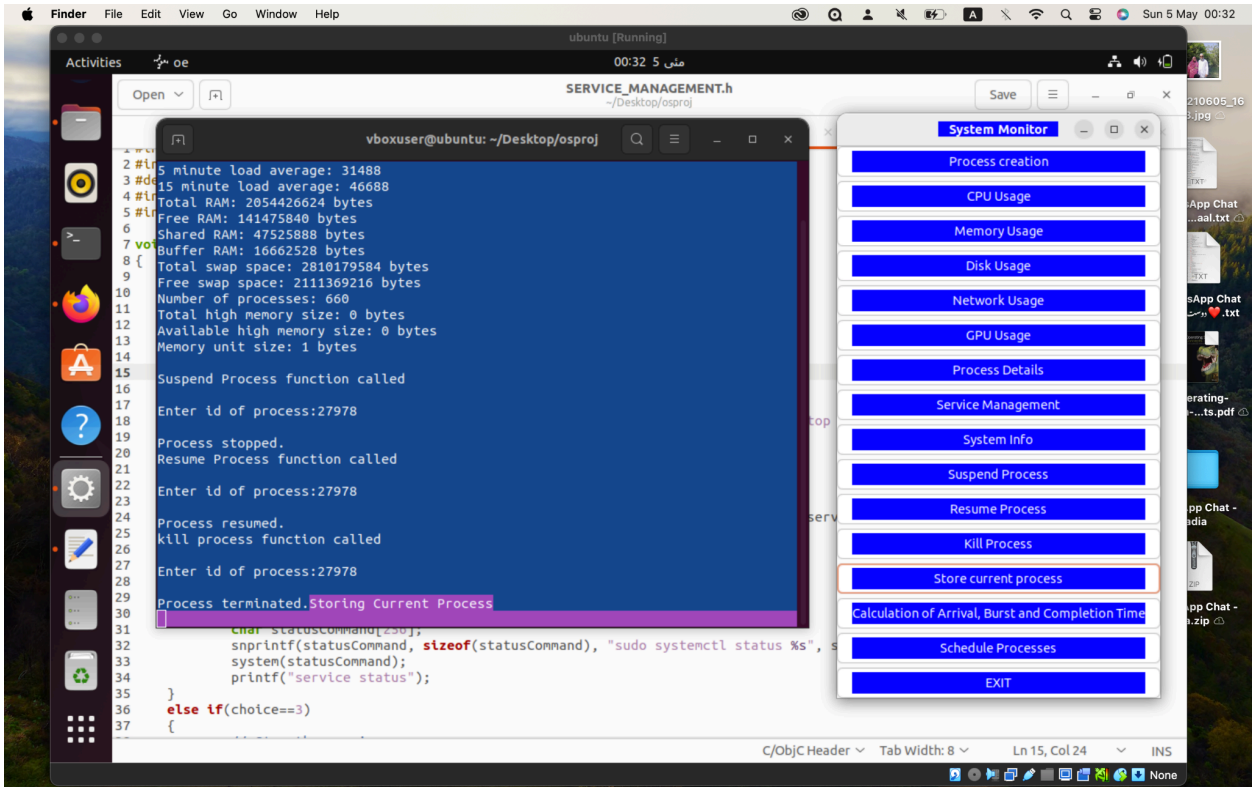
Project folder contains files



System information function called



suspend , resume or kill a process



➢ FUTURE WORK

Future work may involve adding additional functionalities, improving user interface design, optimizing resource utilization, and enhancing the overall performance and reliability of the system monitor tool.

➢ CONCLUSION (SUMMARY & DISCUSSION)

In conclusion, the project aims to develop a comprehensive system monitoring tool with a user-friendly GUI interface. By leveraging GTK and C programming language, the program provides functionalities for monitoring system resources, managing processes and services, and implementing scheduling algorithms. Further enhancements and optimizations can be explored for future improvements.