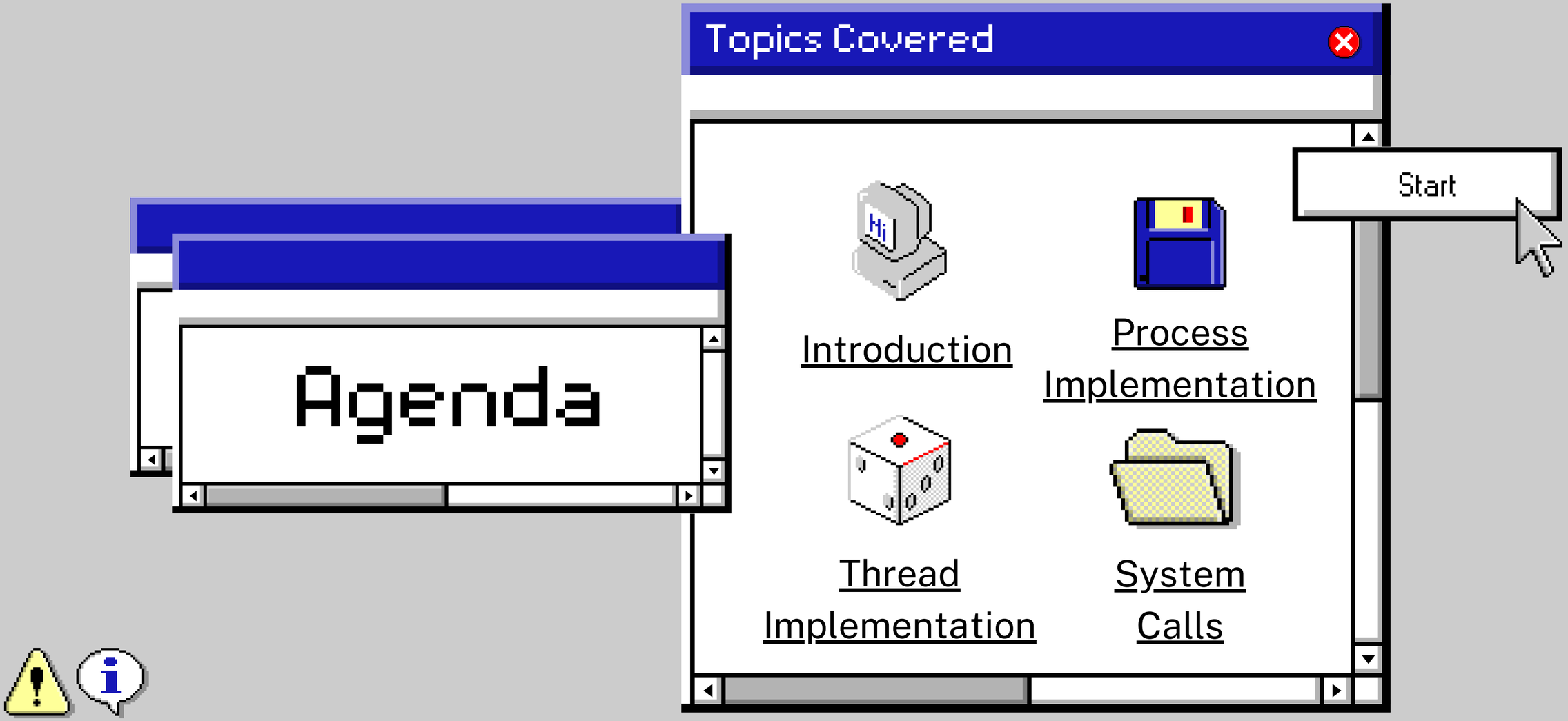
ZORIN OS

A Case Study Project

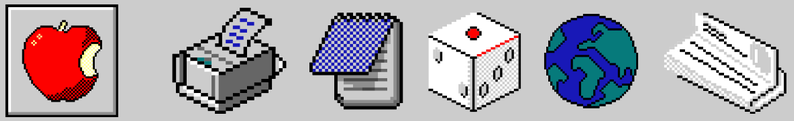
Fatima Farooq 2023006109
Alya Alzaabi - 2022005560
Rabaa Alshbib 2022005690
roodh alblooshi 2022005691
Lana Zanneh 2022005620
Dema Ammar Al-sos 2021004885

11:11PM


Agenda
Topics Covered
Start
Introduction
Process Implementation
Thread Implementation
System Calls

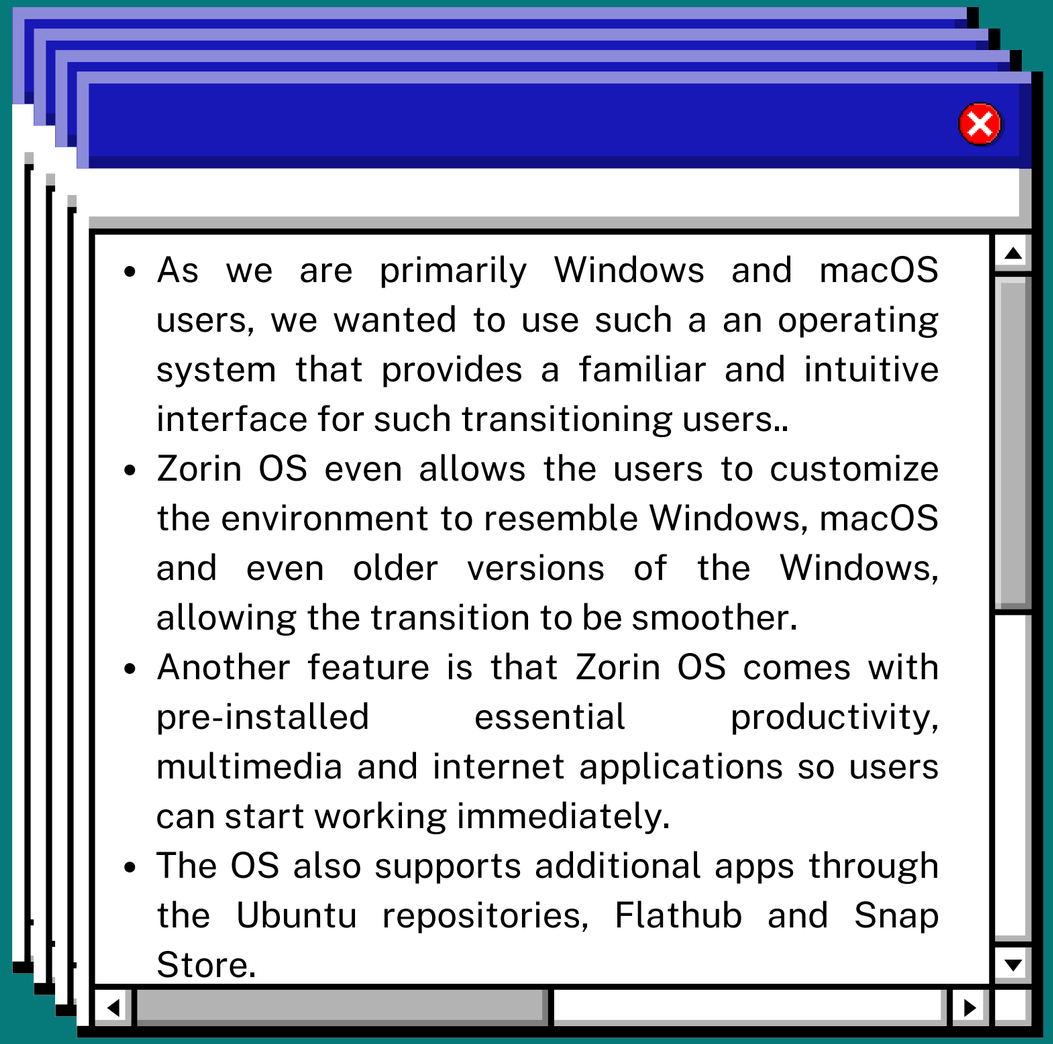
INTRODUCTION TO ZORIN OS?

- Zorin is built on stable Ubuntu Long Term Support Base (LTS) which was launched in 2009 by the Zorin Group in Dublin, Ireland.
- Zorin is a user-friendly Linux distribution which is designed to serve as a convincing alternative to Windows and macOS, allowing the options of both free and paid.
- Its primary goal is to make advanced computing accessible to users by combining the power and security of Linux that provides an interface which is similar for use to other operating systems.
- Our case study features the latest Zorin OS 17.3 release version in the form of core.



Back to Agenda Page

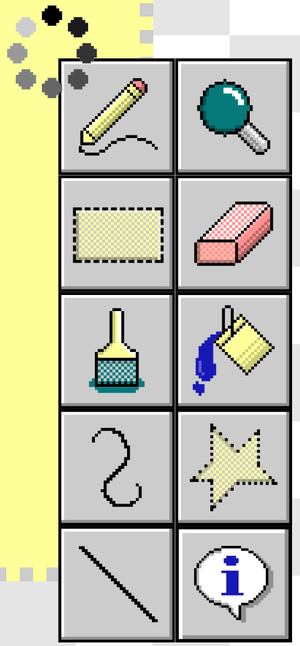
WHY ZORIN OS?



- As we are primarily Windows and macOS users, we wanted to use such a an operating system that provides a familiar and intuitive interface for such transitioning users..
- Zorin OS even allows the users to customize the environment to resemble Windows, macOS and even older versions of the Windows, allowing the transition to be smoother.
- Another feature is that Zorin OS comes with pre-installed essential productivity, multimedia and internet applications so users can start working immediately.
- The OS also supports additional apps through the Ubuntu repositories, Flathub and Snap Store.

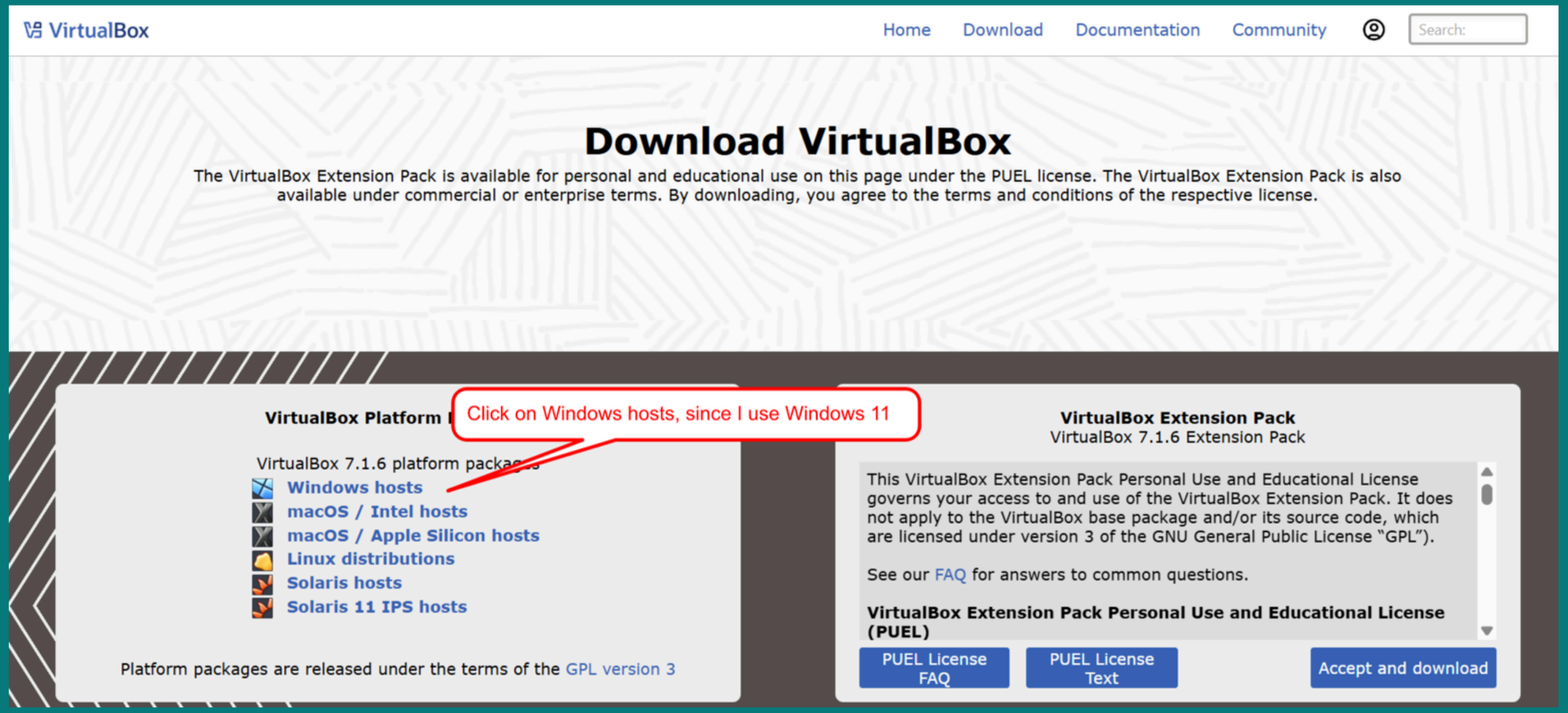
Back to Agenda Page

Steps for Installing Zorin OS

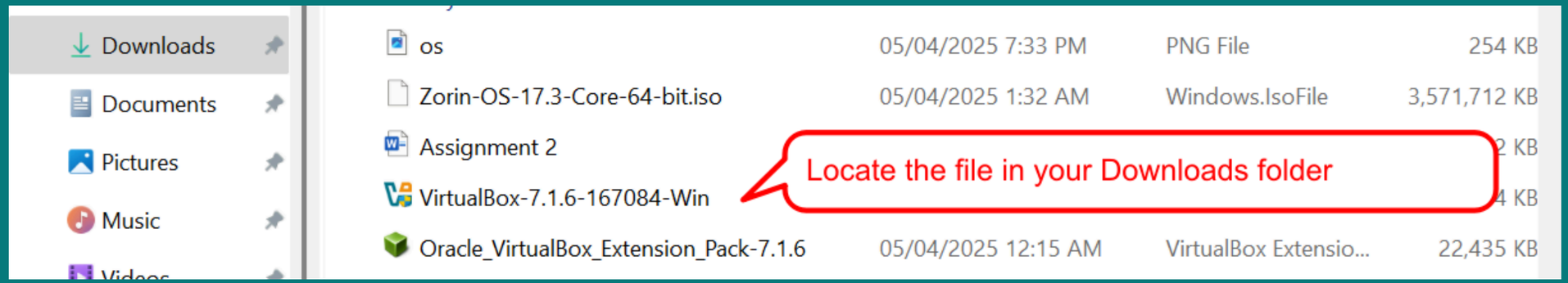


Back to Agenda Page

1. Set Up a Virtual Machine

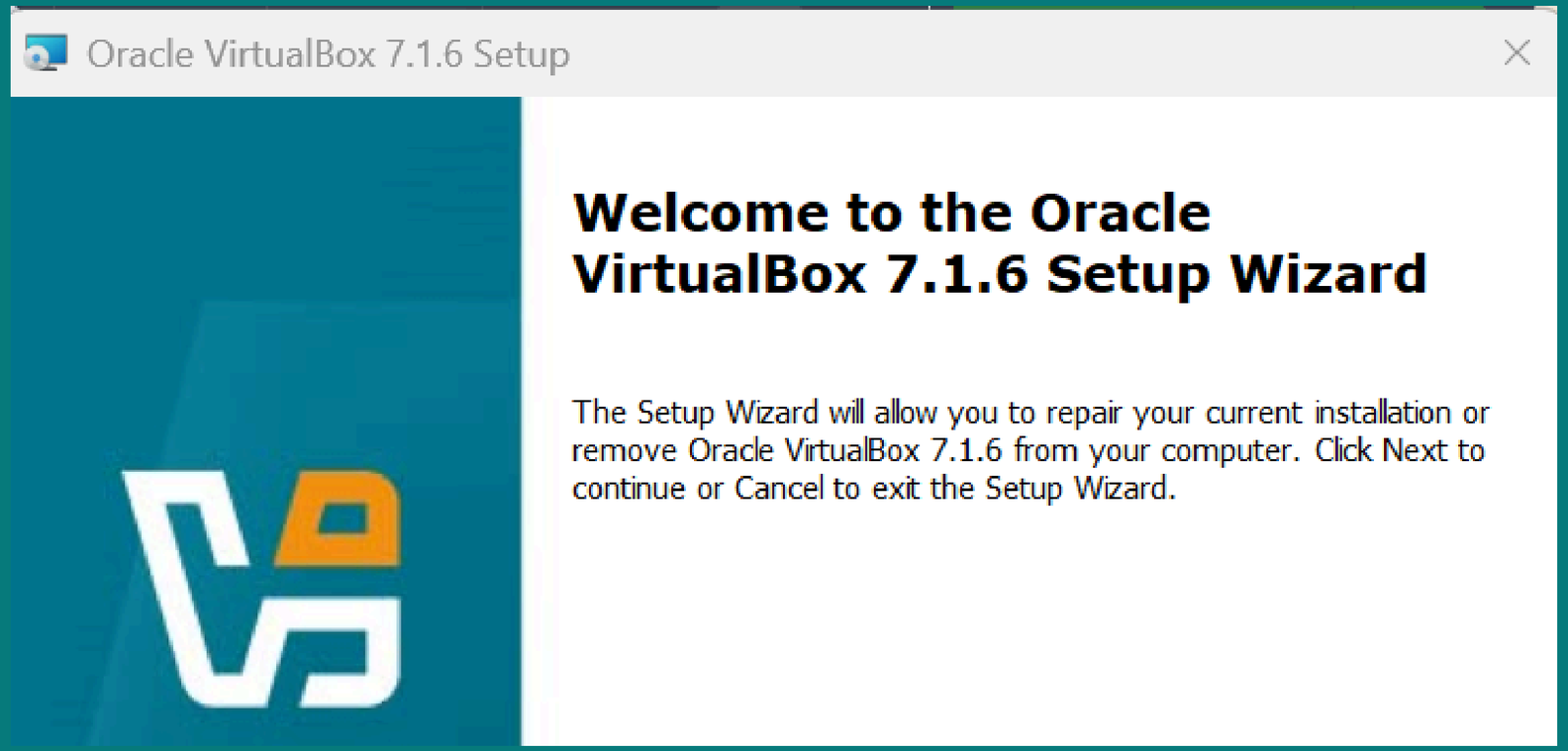


Back to Agenda Page



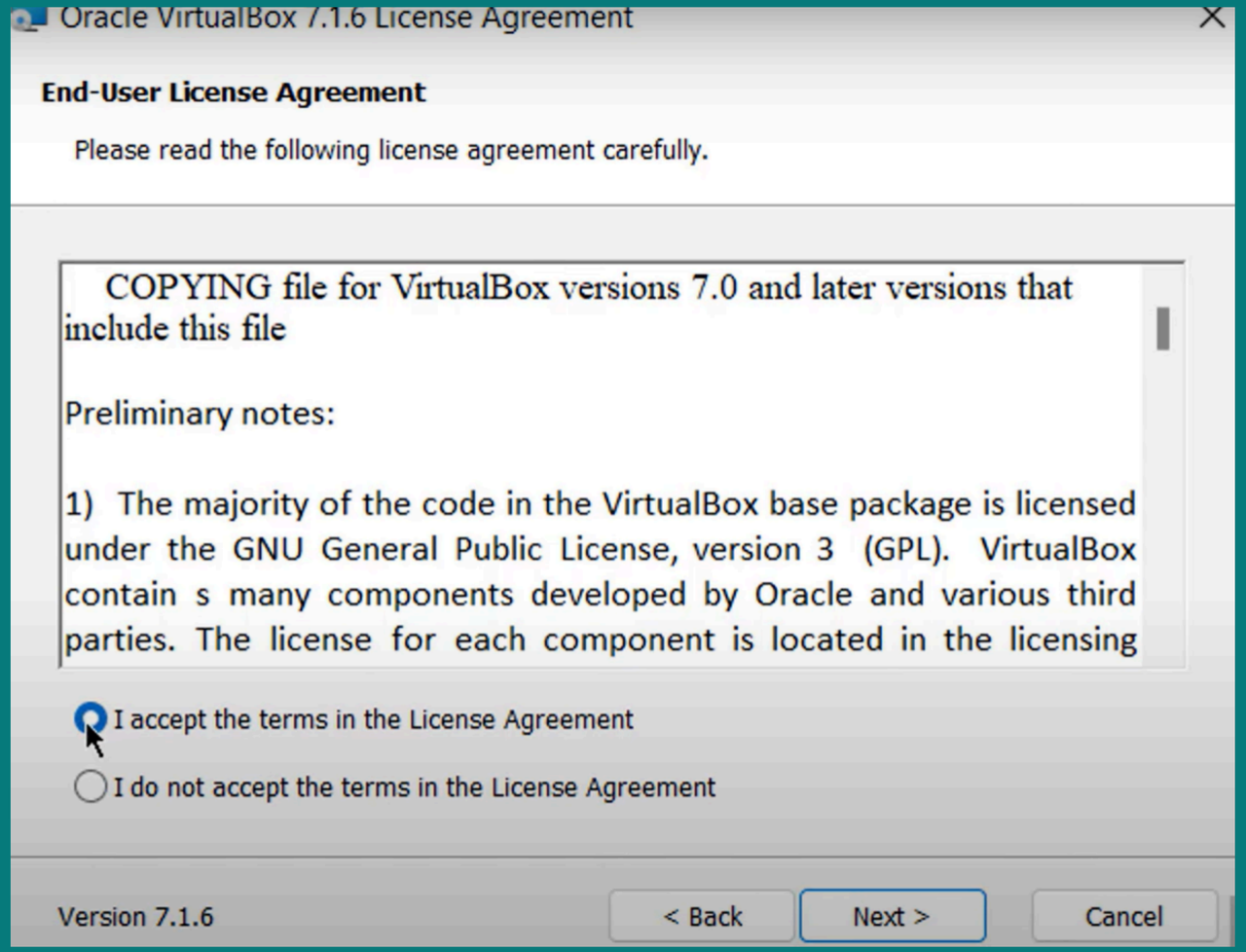
Click on the file

Back to Agenda Page



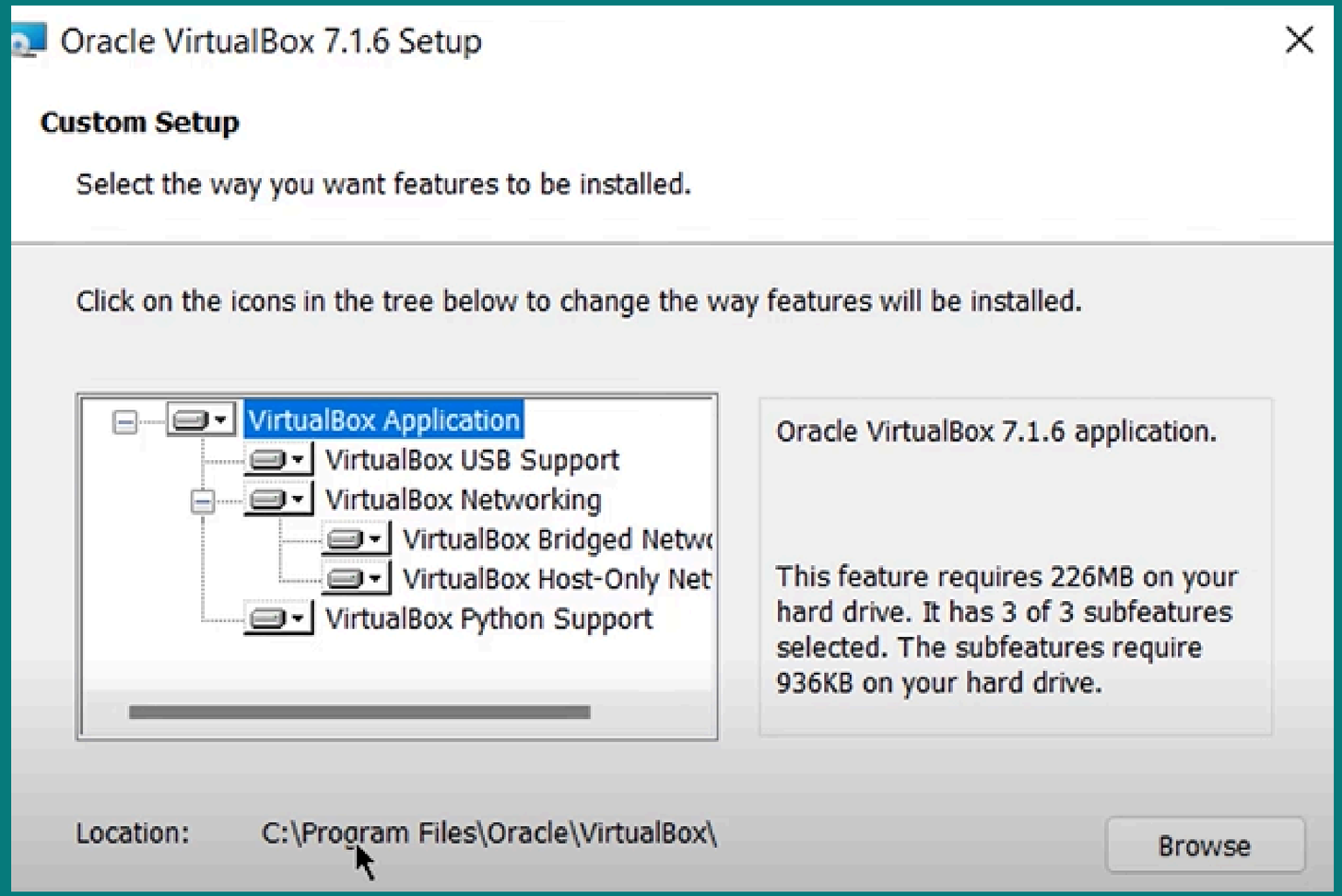
Set up the Oracle VirtualBox

Back to Agenda Page



Accept the License Agreement

Back to Agenda Page

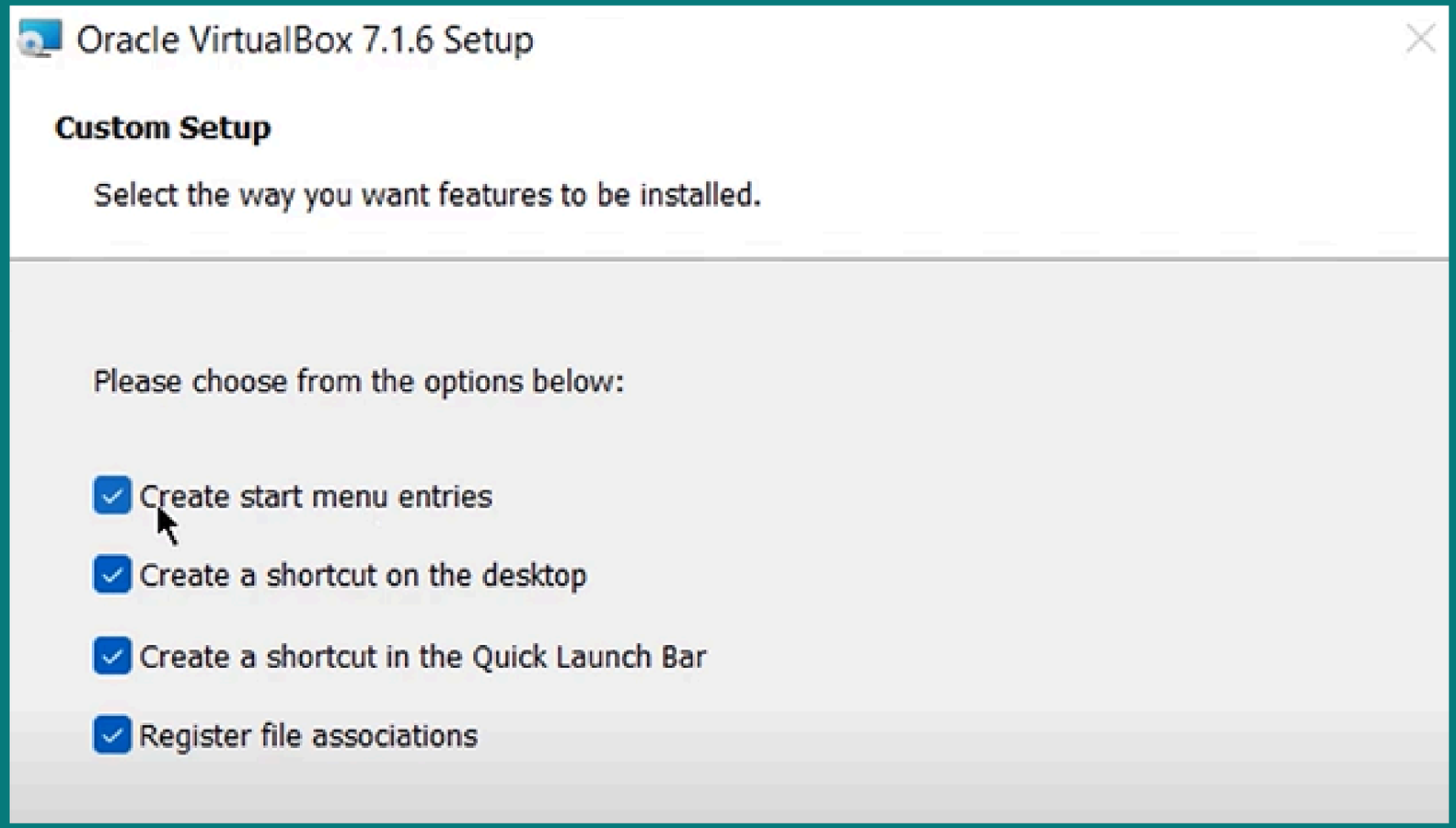


Keep Default Settings

Back to Agenda Page

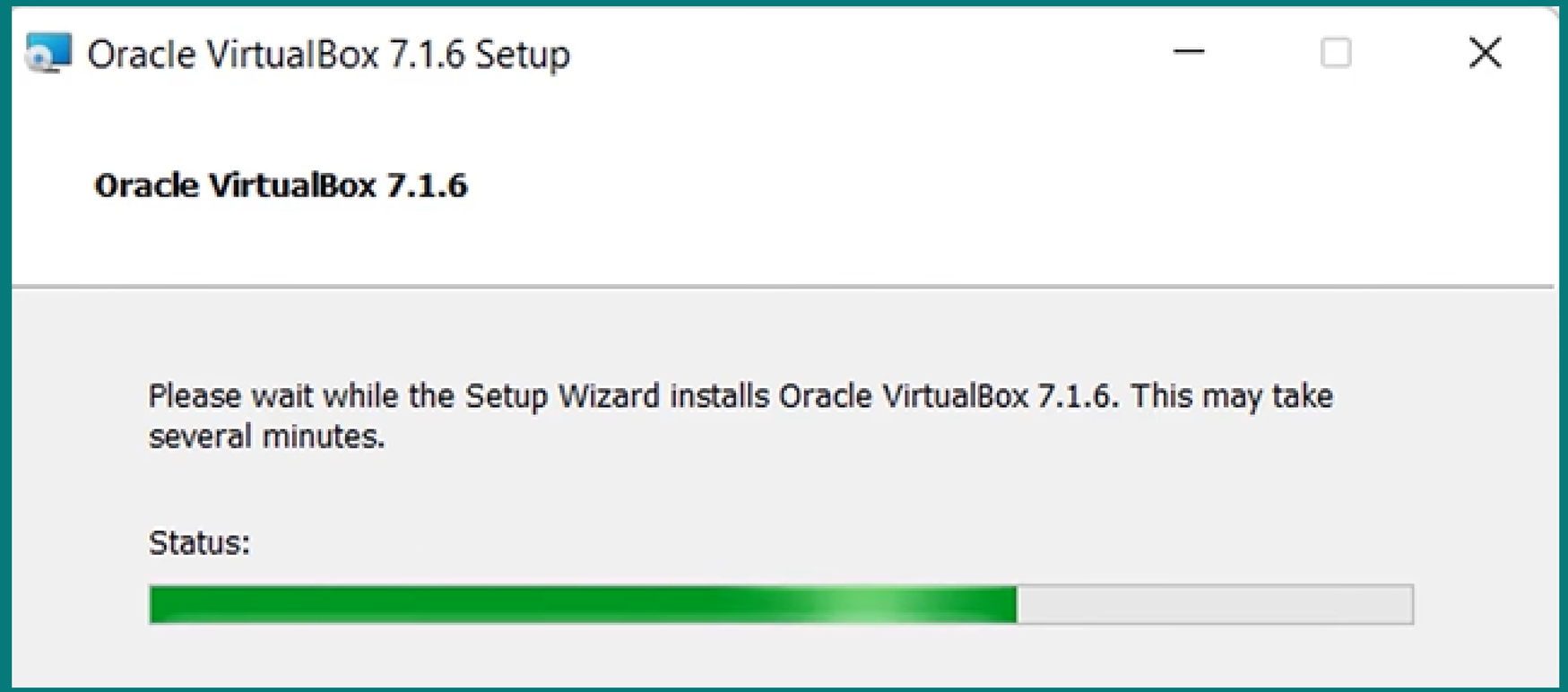
Proceed with Installation

Back to Agenda Page



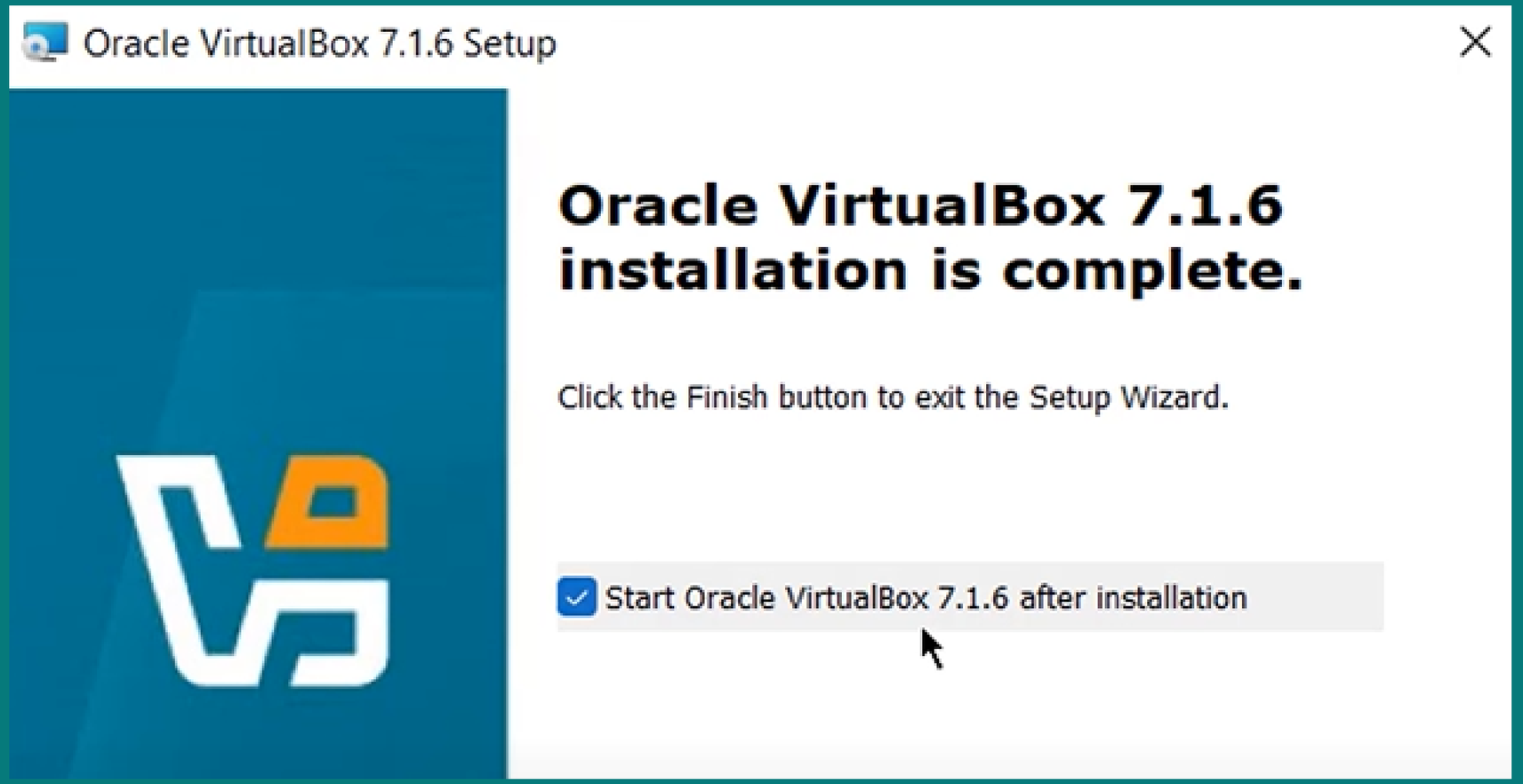
Choose based on your preference

Back to Agenda Page



Wait for it to install

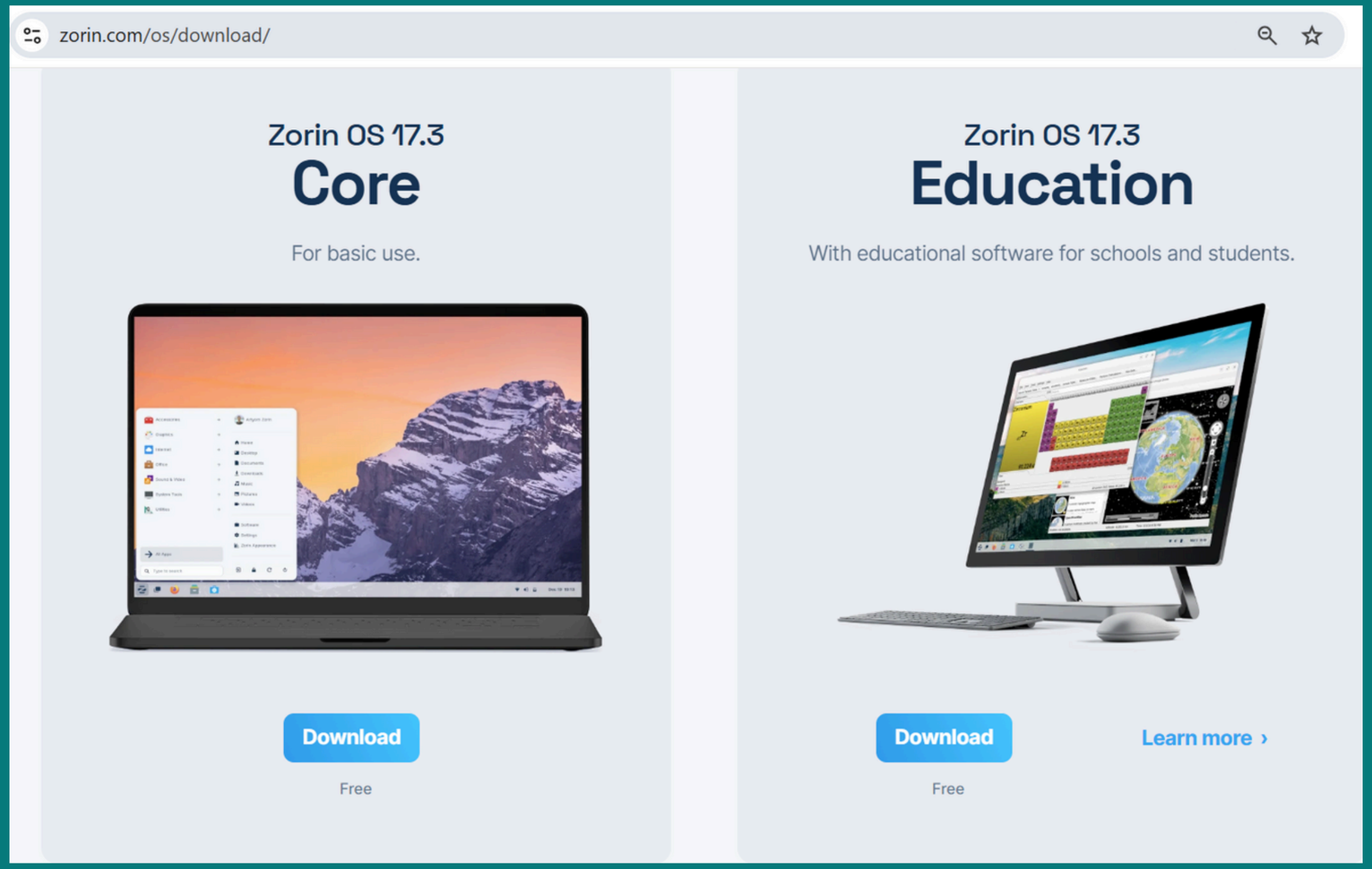
Back to Agenda Page



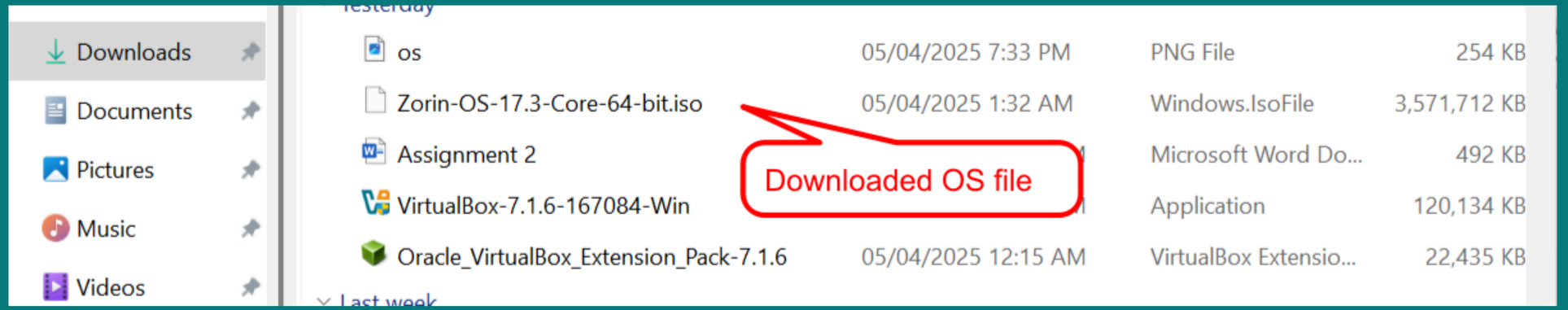
Once it is finished, run it

Back to Agenda Page

2. Install Zorin OS (Core Version)

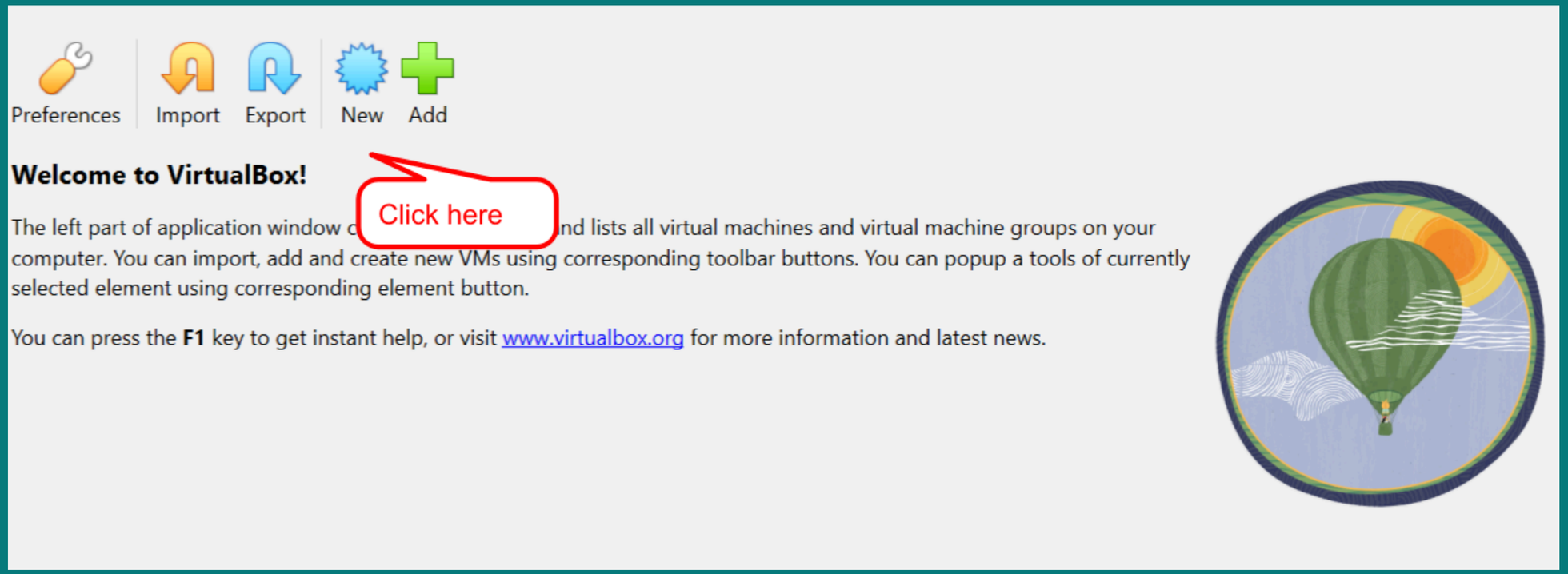


Back to Agenda Page



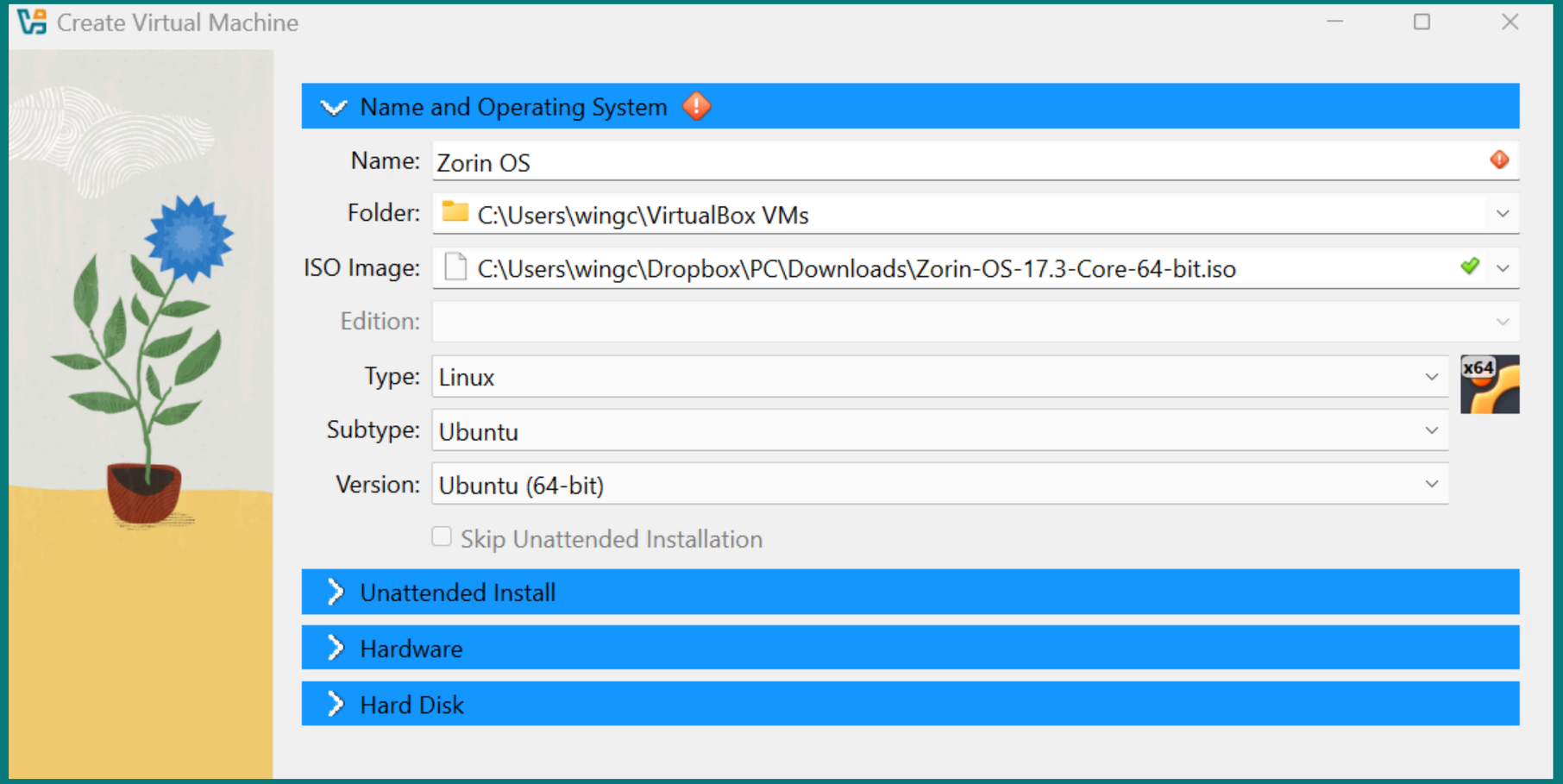
Locate the .iso file in your downloads folder

Back to Agenda Page



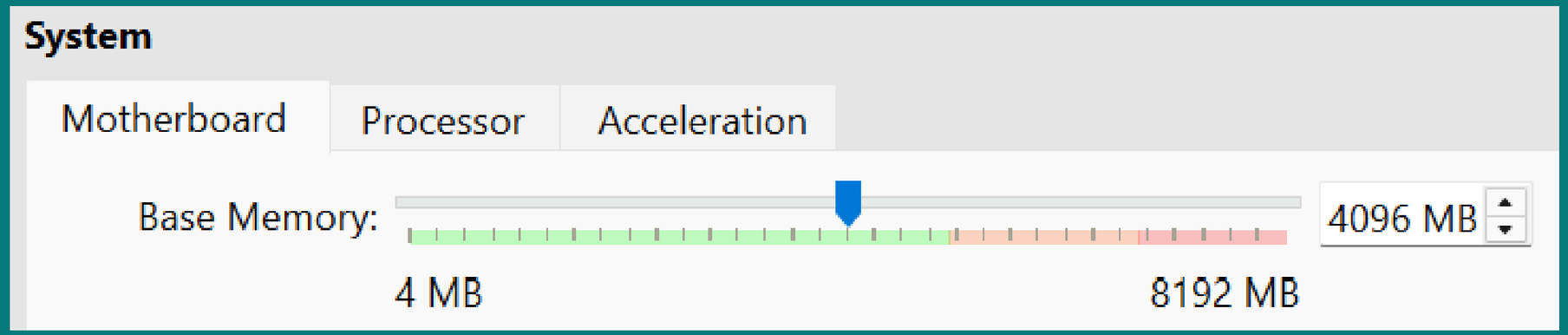
Create a new Virtual Machine (for Zorin OS) by clicking on new

Back to Agenda Page



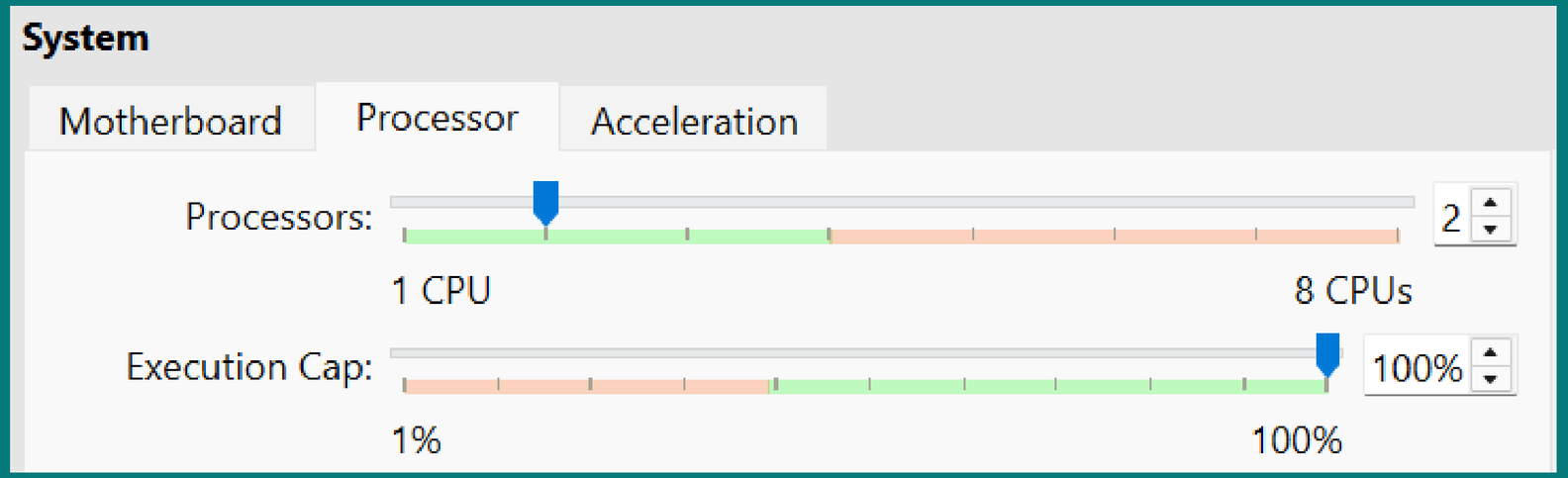
Will automatically detect it as Ubuntu (64-bit)

Back to Agenda Page



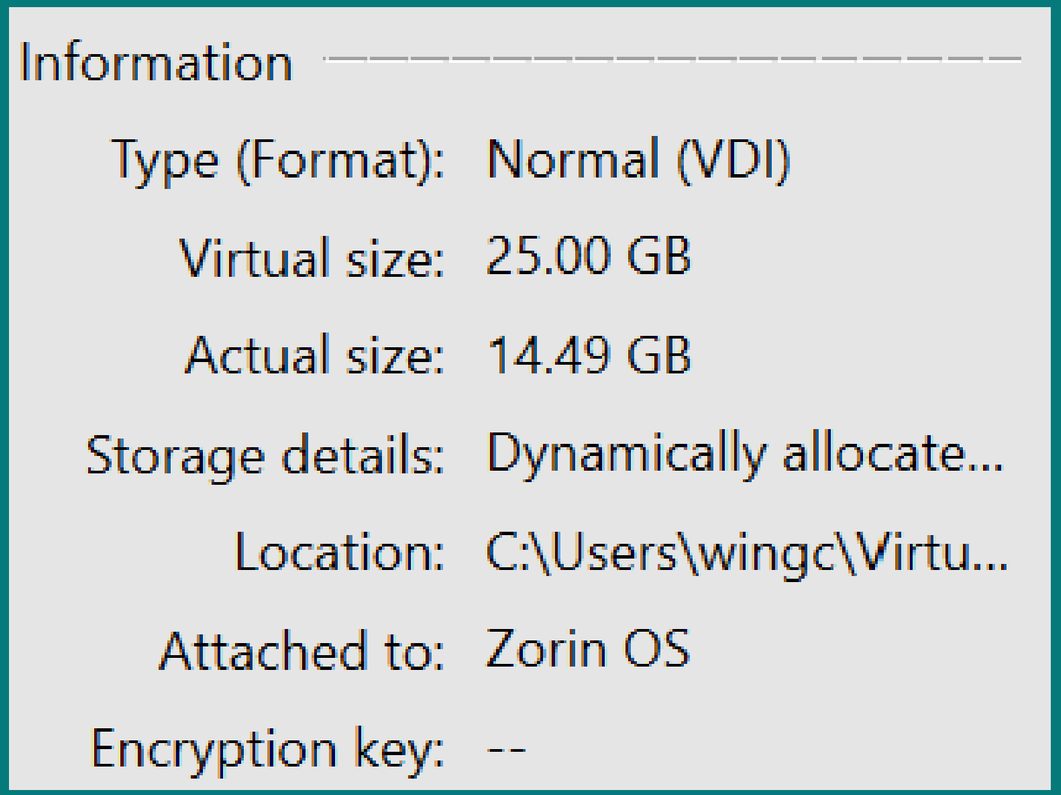
Set the Base Memory to 4GB, since my Laptop has only 8GB RAM, so ensures smooth performance for both VM and laptop.

Back to Agenda Page

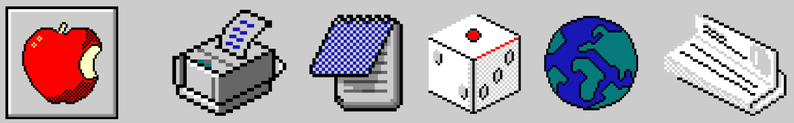


Set the processors to two, so it balances the performance without slowing down the laptop

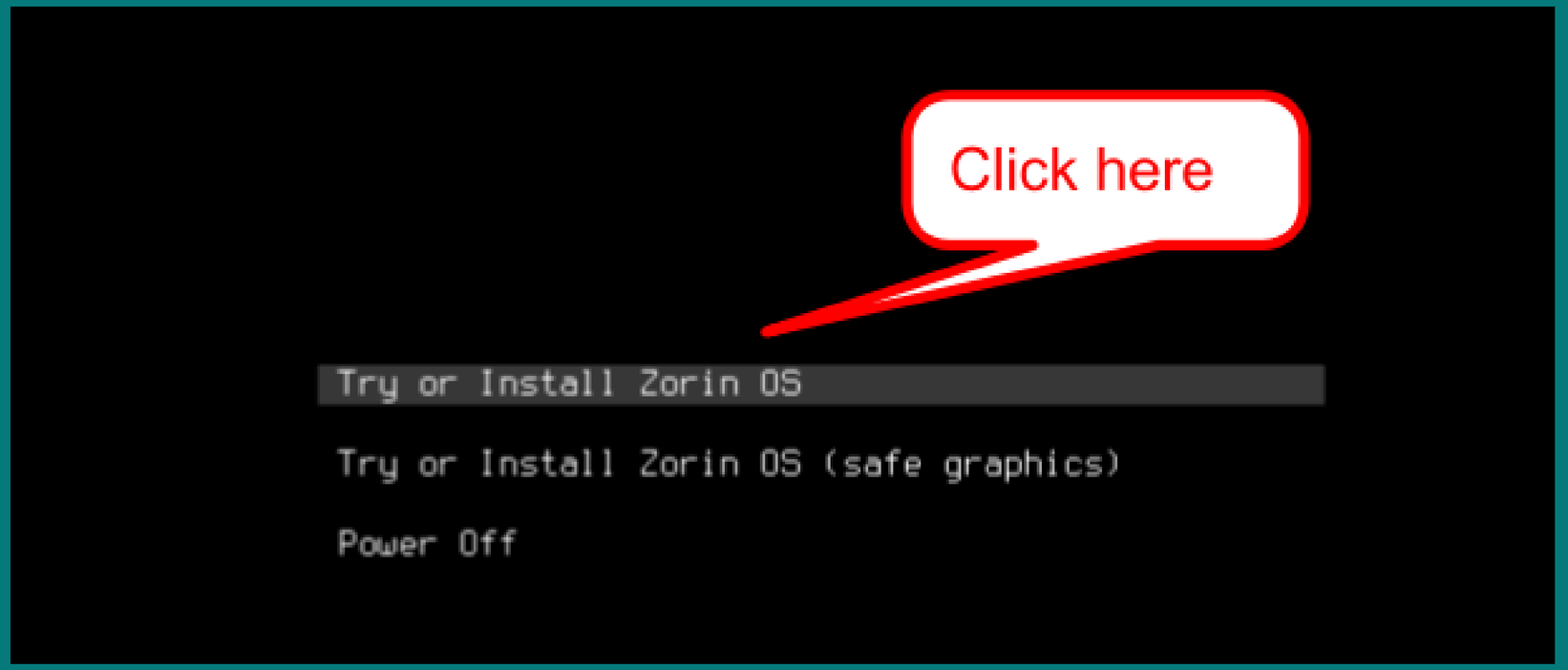
Back to Agenda Page



Zorin requires minimum of 20GB, for flexibility, choose 25GB.

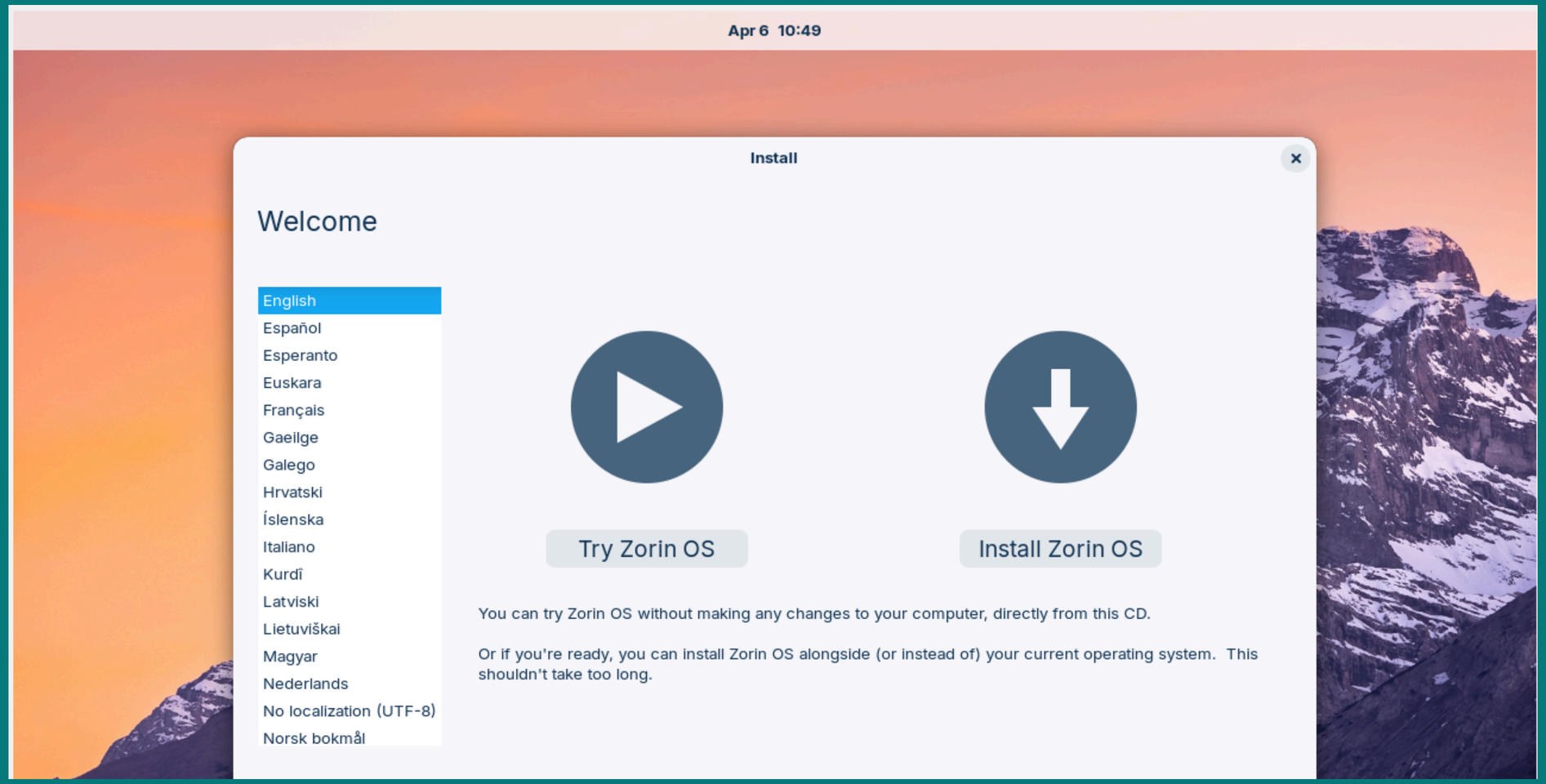


Back to Agenda Page



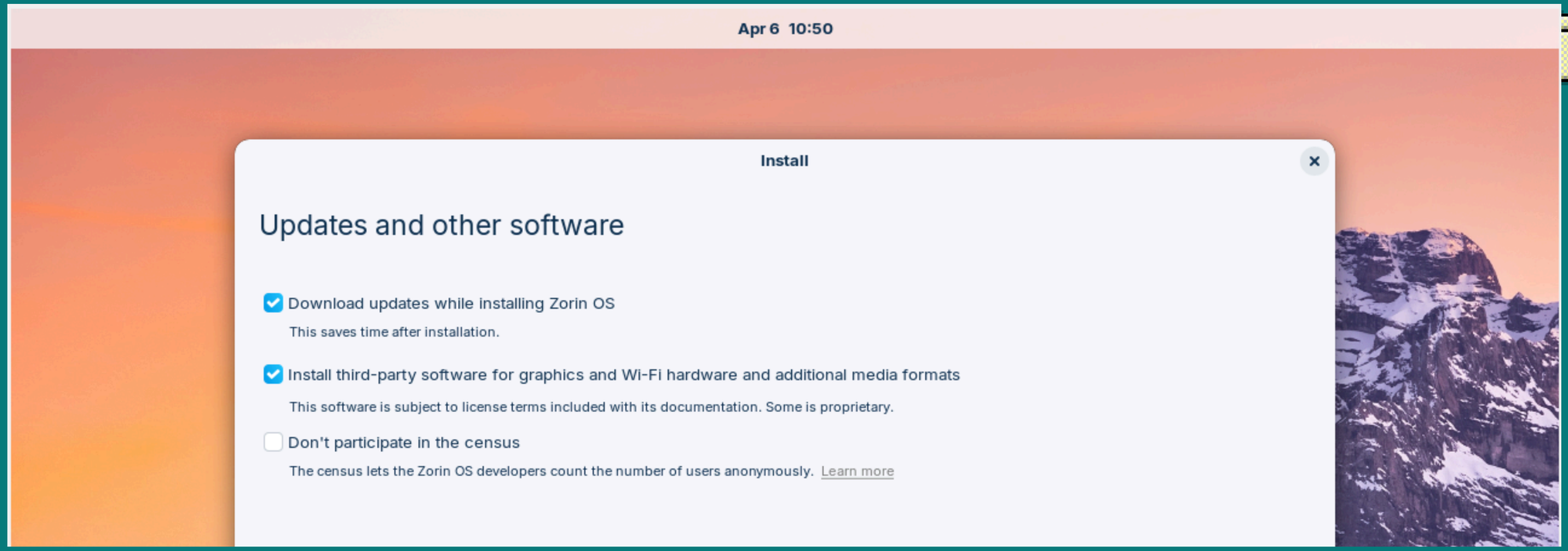
After you have finished your hardware specifications, you are led to this menu

Back to Agenda Page



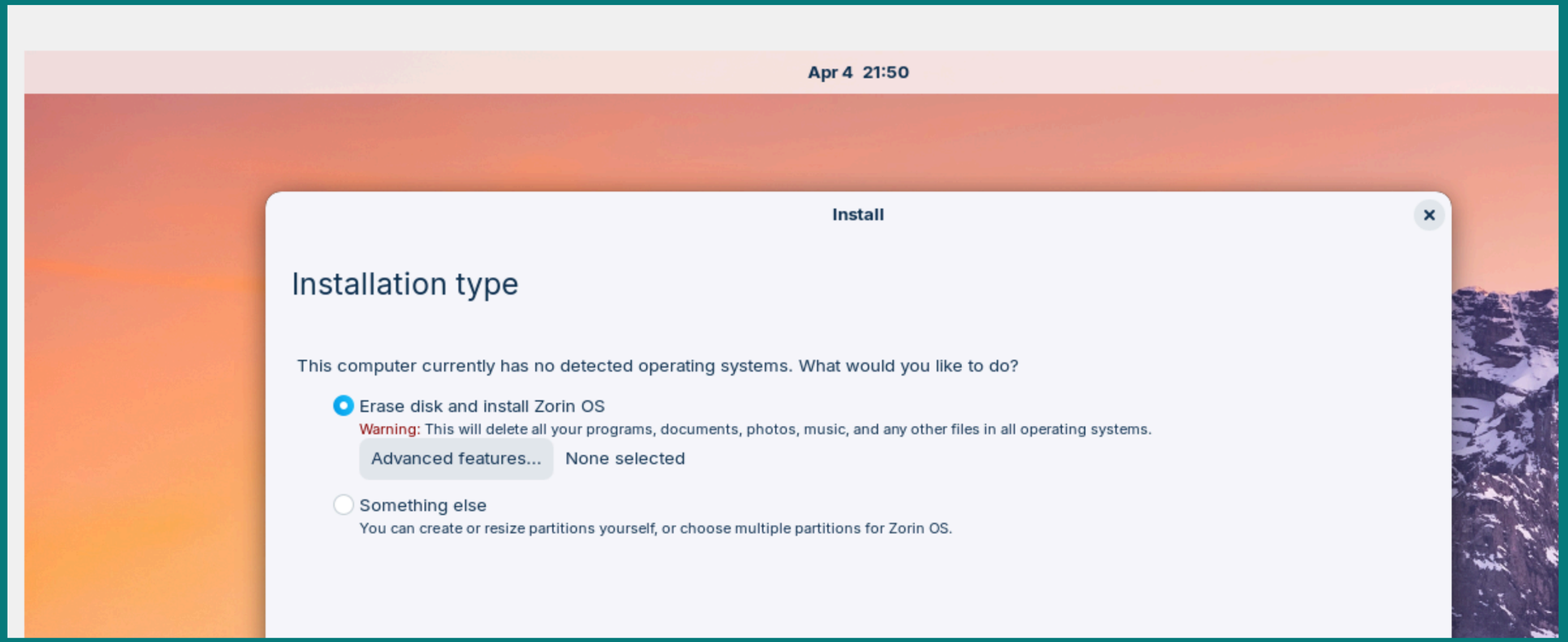
Install Zorin OS

Back to Agenda Page



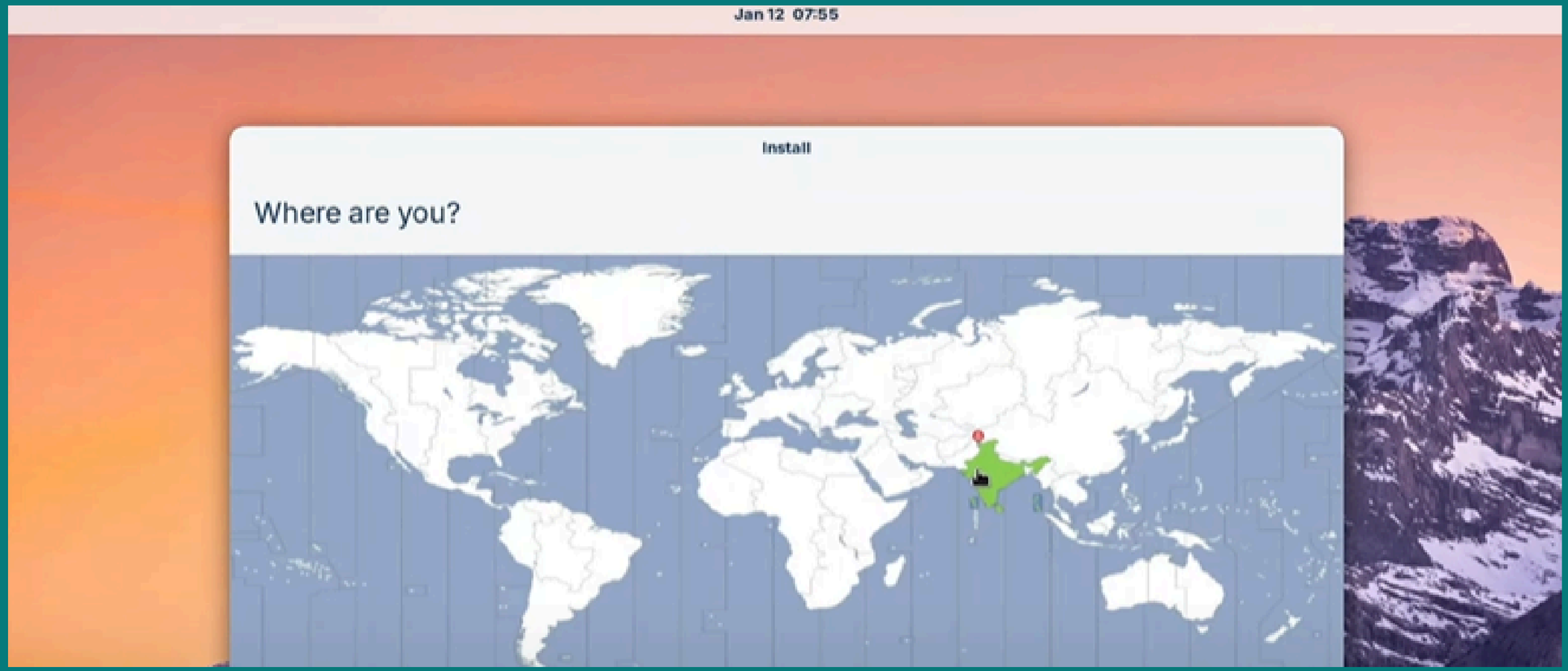
Choose based on your preference

Back to Agenda Page



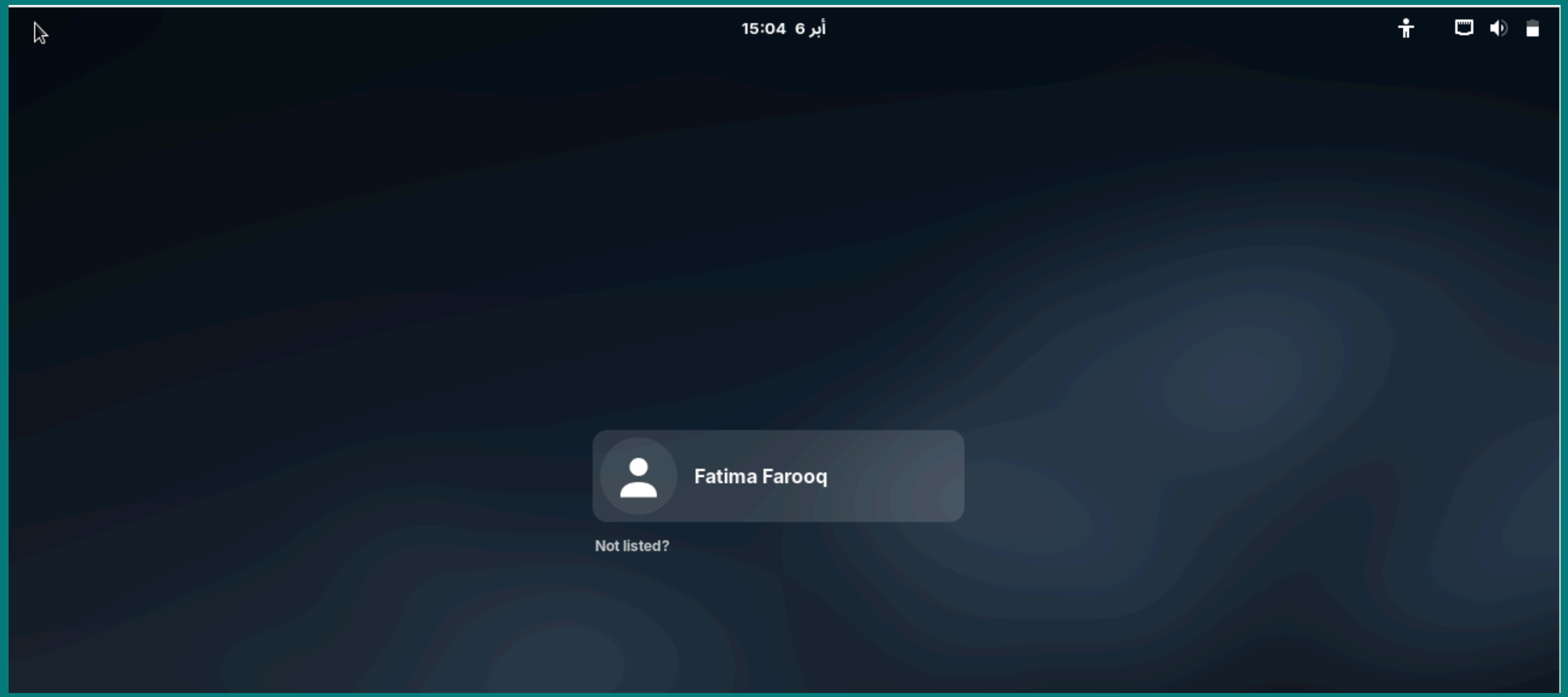
Choose Erase Disk because we do not have an existing OS on VM

Back to Agenda Page



Choose your location

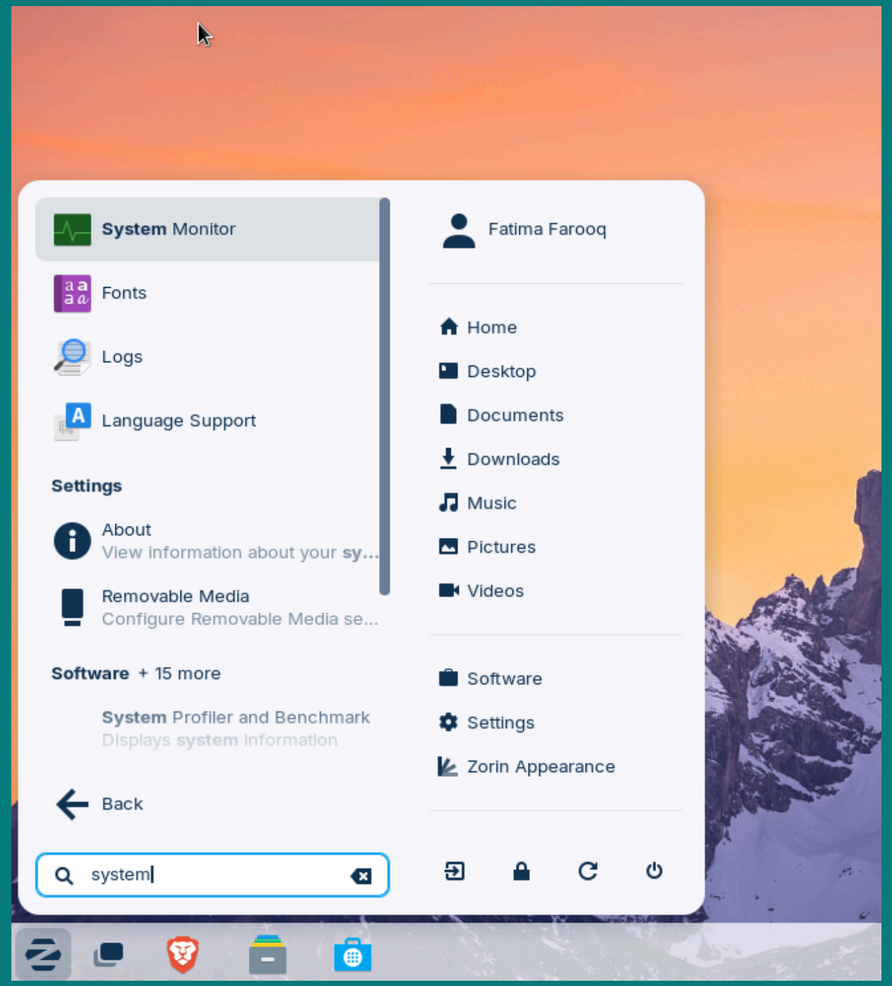
Back to Agenda Page



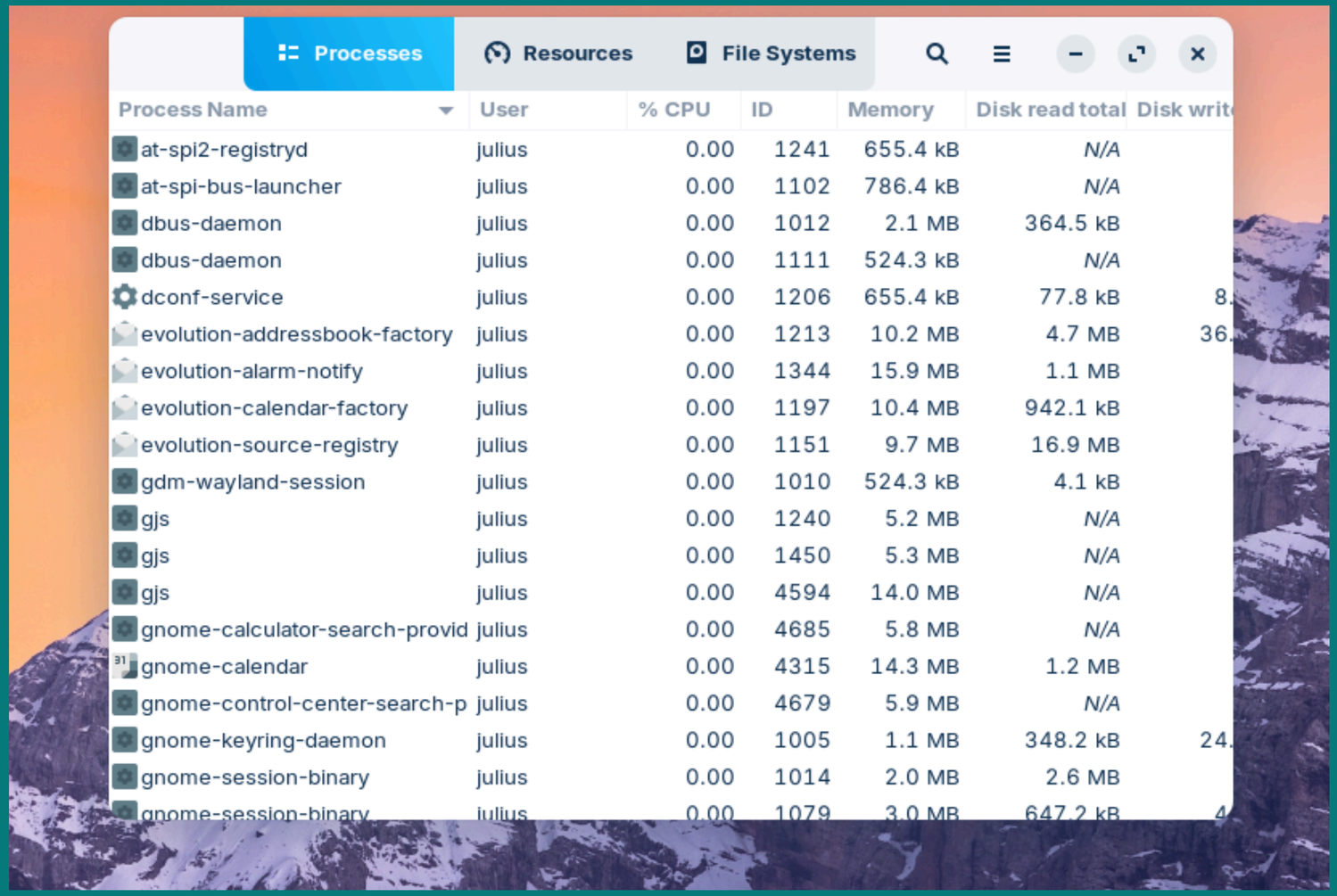
Finally, login

Back to Agenda Page

3. Install System Monitoring Tools

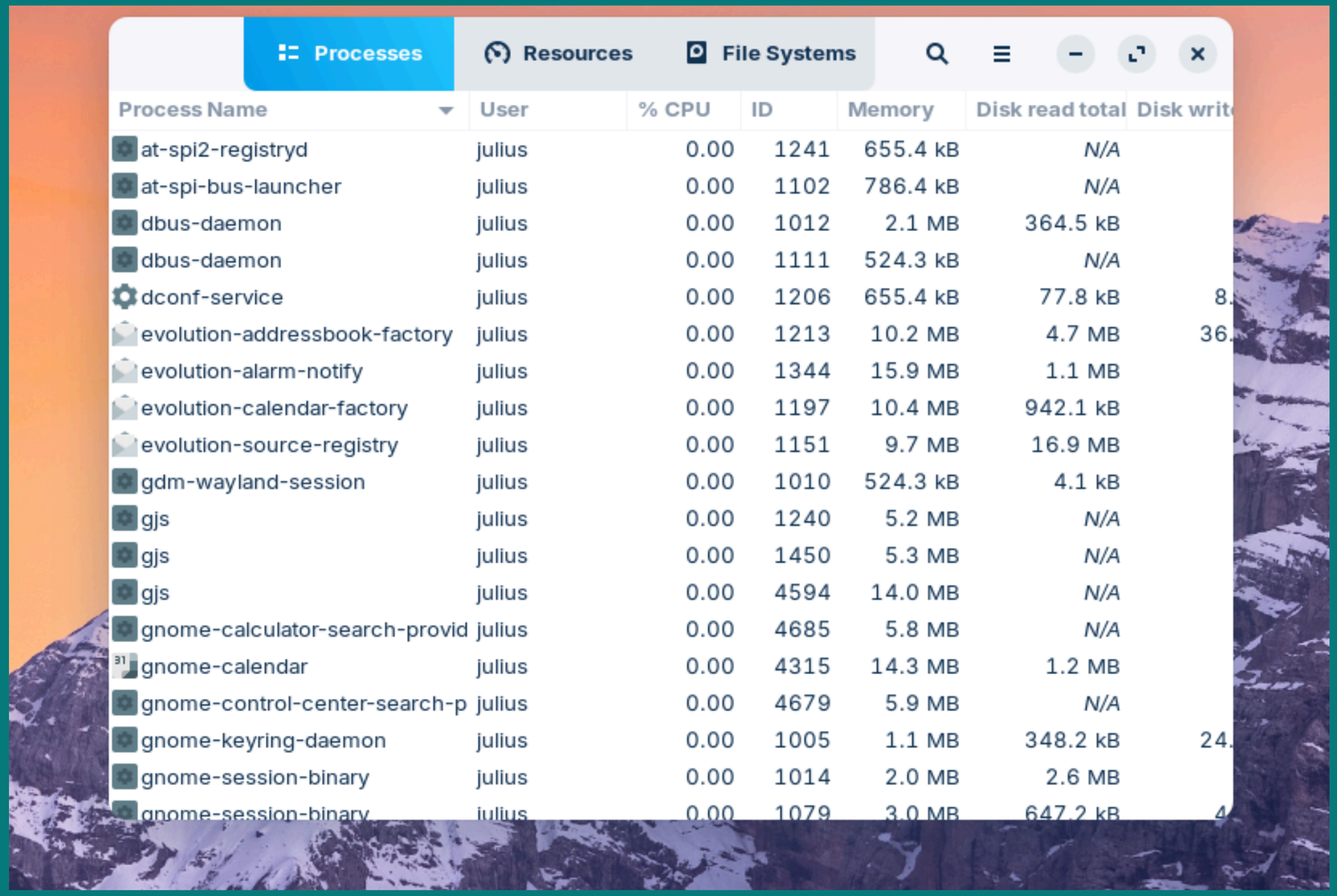


Already installed by default



Opens the current running processes

Back to Agenda Page



Opens the current running processes

Back to Agenda Page

OVERVIEW OF THE RUNNING PROCESSES

System and Background Services

•dbus-daemon: this process aids the inter-process communication (IPC) framework for messaging and coordination utilized by many other processes.
• NetworkManager: it allows the managing of networking interfaces and ensuring connectivity, while making sure that the system has an established connection and can adjust to changes caused by network availability.
• systemd: This is the system and service manager of the Linux Operating Systems.
• cupsd: This process is responsible for handling printing services.
• Audio Server (pulseaudio or pipewire): This handles the stream mixing of audio, its routing and volume control for both the sounds of the system and the user applications.
• avahi-daemon: This provides a service discovery on a local network.
• sshd: This daemon Secure Shell (SSH) enables remote access.

Back to Agenda Page

OVERVIEW OF THE RUNNING PROCESSES

GNOME/Desktop Components

• gnome-shell: This is the core process of the GNOME desktop environment that takes care of the graphical interface, window management and the overall desktop composition. Because it is largely multi-threaded, it can separate UI rendering, input entry and other desktop functions.
• gnome-session: This process manages and deals with the GNOME session.
• gnome-settings-daemon: This process allows the implementation of different settings for the GNOME desktop.
• gnome-shell-calendar-server: This allows us to manage the calendar events in the GNOME shell.
• gnome-software: This is an application for Software Management.
• GNOME Settings Daemons (gsd-*): These incorporate a list of dedicated daemons which handle particular roles.
goa-daemon (GNOME Online Accounts Daemon: This makes the integration and synchronization of online ac-
counts like Google, Facebook with GNOME applications which are usually running in user space.

Back to Agenda Page

OVERVIEW OF THE RUNNING PROCESSES

User Applications

•Evolution: This includes email, calendar, contacts and tasks which acts as a frontal implementation for management for personal information.
• evolution-data-server: For purposes of Evolution, it runs in the background to save and manage the data such as email, calendar and contacts.
• evolution-calendar-factory / evolution-addressbook-factory: By these processes, we can run and handle the required parts for calendars and contacts from the Evolution.

Back to Agenda Page

Resource Utilization

CPU

Most processes represent a near-zero CPU usage, which means that they are idle or in a waiting state, which is supportive that is because, I was not running any programs such as web browsers, or media players on the laptop at that moment. Processes like gnome-shell and gnome-settings-daemon represent a significant amount of CPU usage, however it is a continuous and low activity, which means that they are functioning to perform tasks that needs computational power.

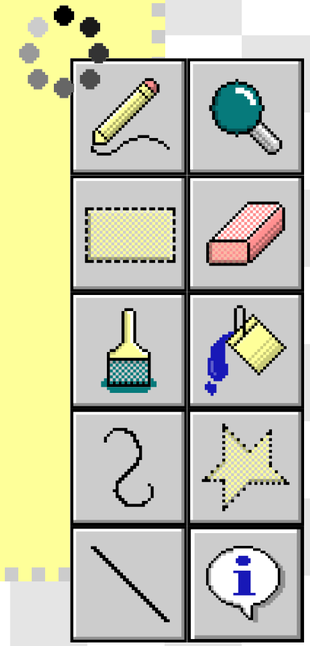
Memory Footprint

There is a detectable change in the memory allocation. Lightweight Daemons require a minimum amount of memory such as a few megabytes. Nonetheless, applications that involve user interactions like web browsers or multimedia tools consume a larger memory. Through this distribution, we can highlight the efficiency of memory management so that important processes are placed an emphasis on response while user related requests and applications may have more varied resource demands. We see that gnome-shell uses a significant amount of the memory which is assertive of the fact that the graphical user interface along with other system functionalities needs to be maintained.

Disk I/O

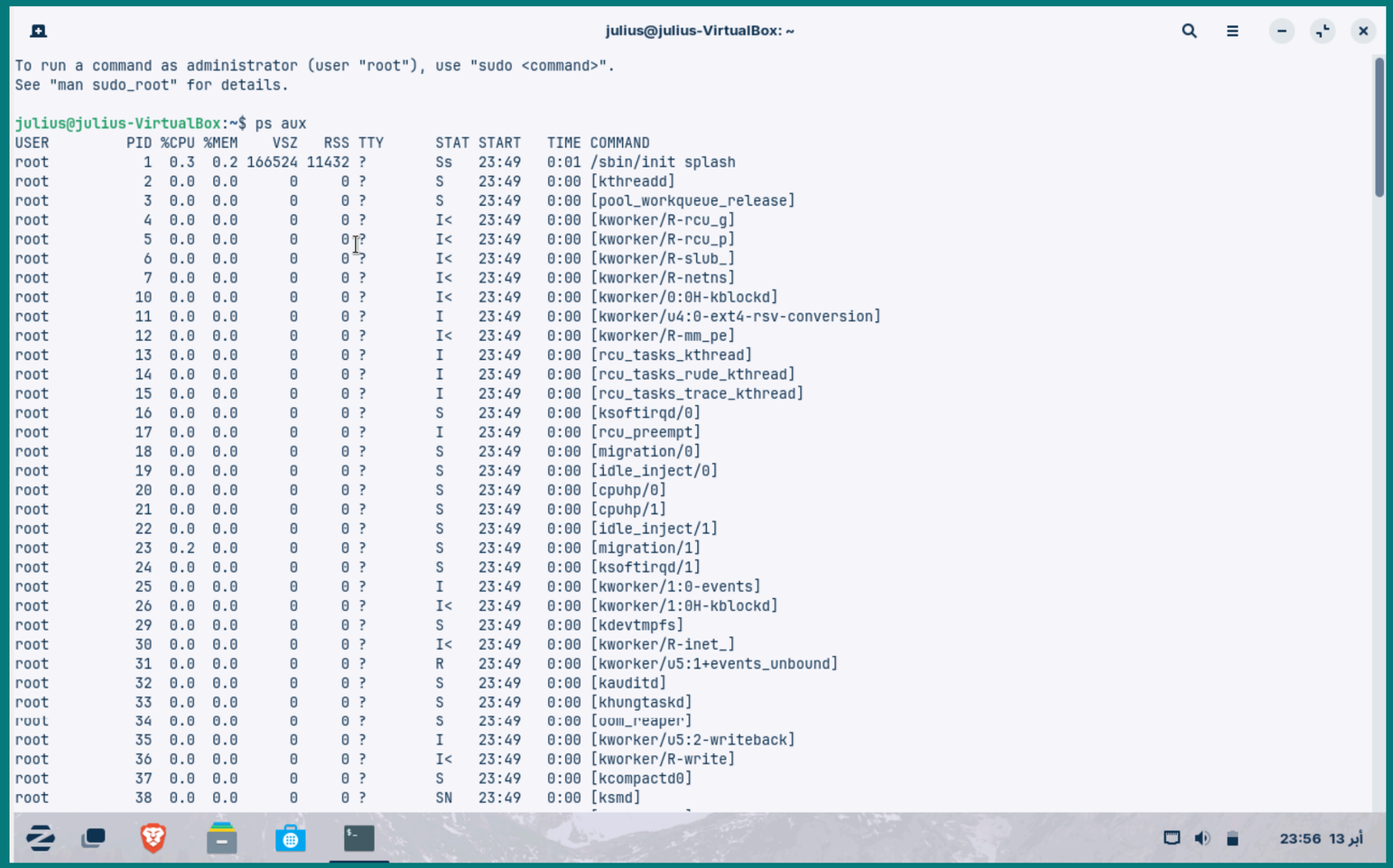
Because of the several number of processes running in the background, these tend to be updated at regular cycles by the monitoring tools, because some processes may not show active disk usage. There can be temporary spikes or inactivity in the disk I/O which can signal which stage the process is at, reading or writing or even loading while performing tasks like file transfers or software updates

Terminal Implementation of Processes

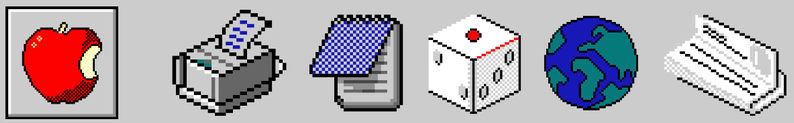


Back to Agenda Page

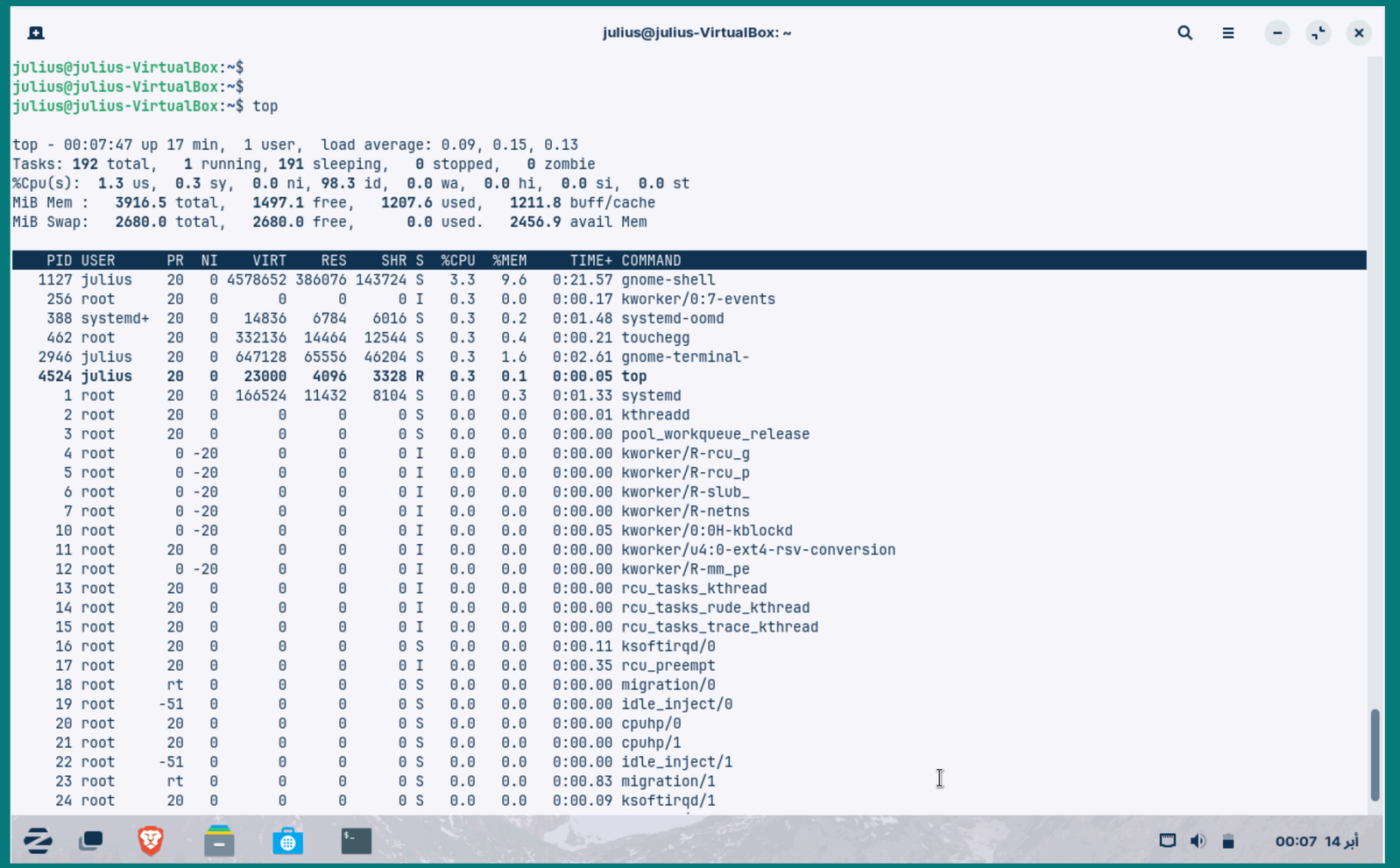
Process Status (ps): ps aux



Shows all running processes in a detailed, user-friendly format, including background ones. Filter with grep: ps aux | grep gnome

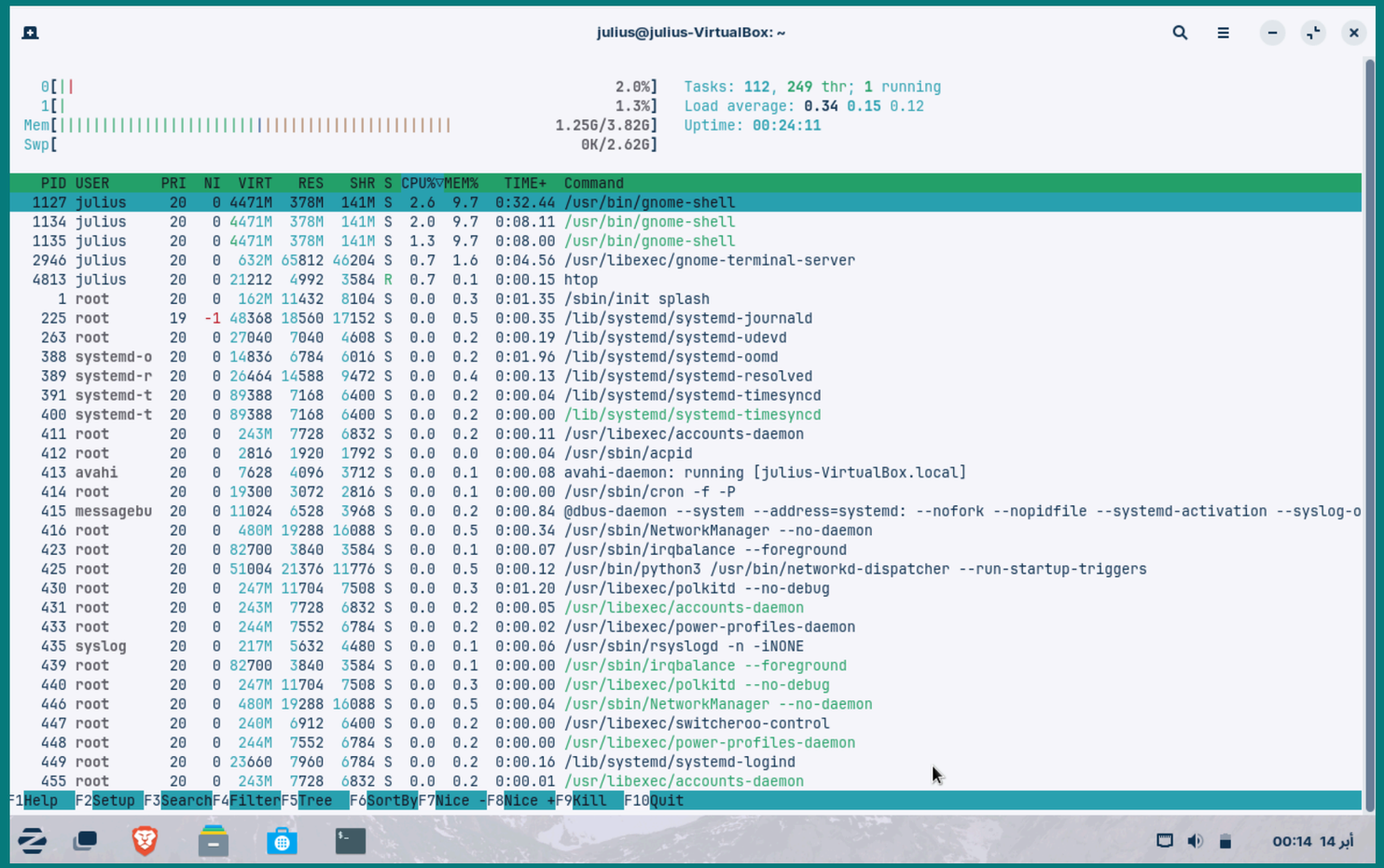


Real-Time Process Viewer:top



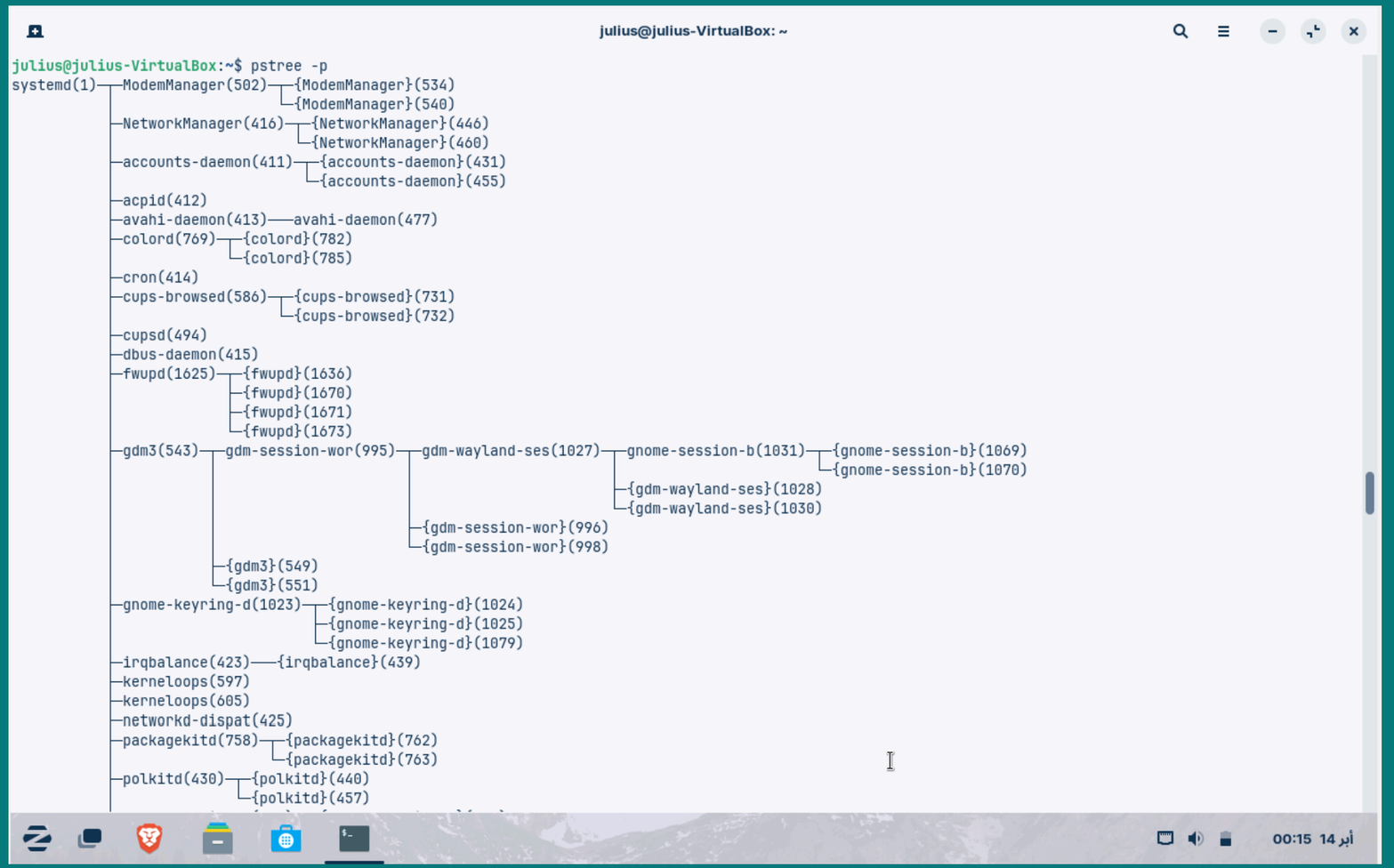
Shows live CPU and memory usage for processes. It refreshes every few seconds.

Improved top Interface:htop



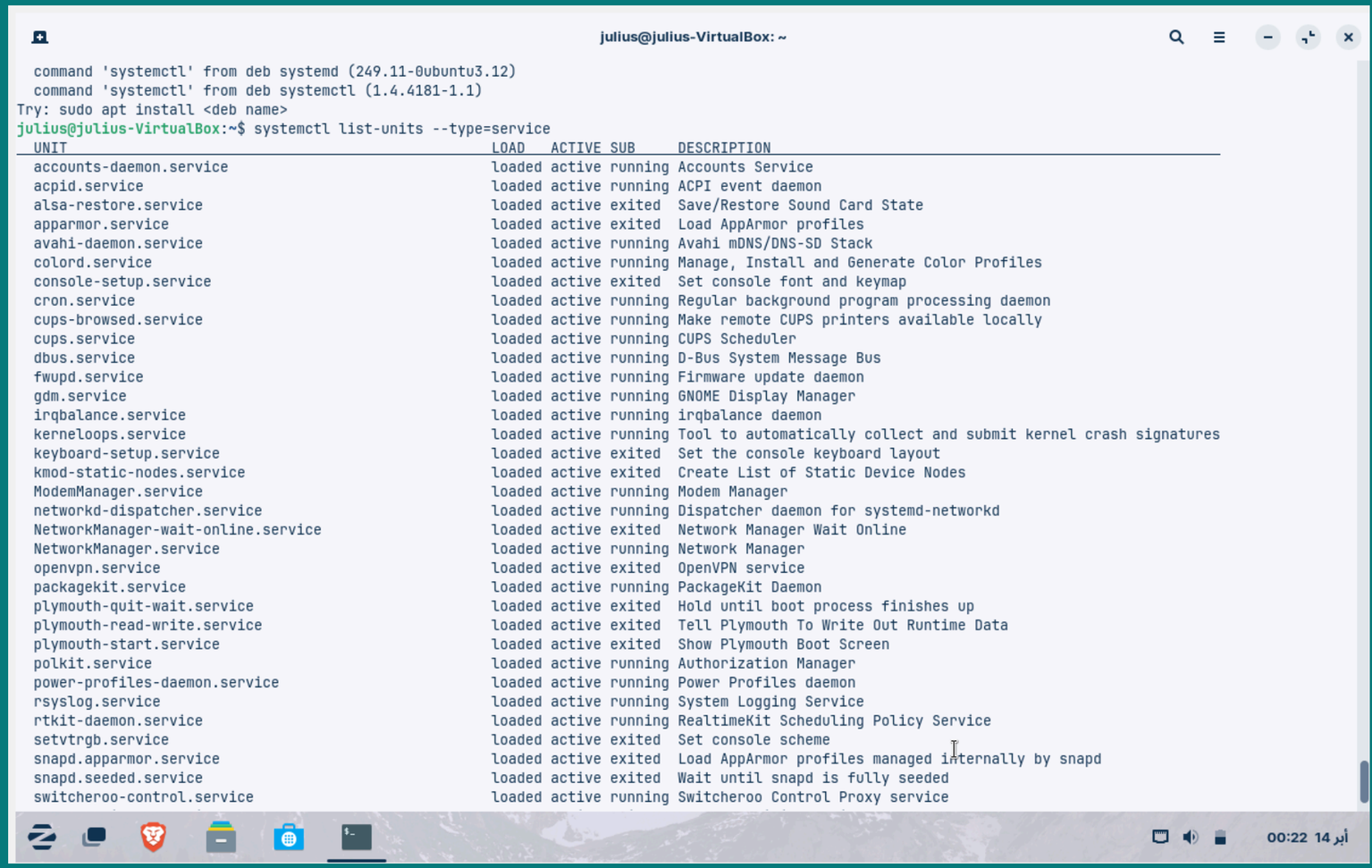
Provides a colorful, interactive process list. It even allows you to scroll, search (/), and sort by CPU/memory.

Process Hierarchy:pstree -p



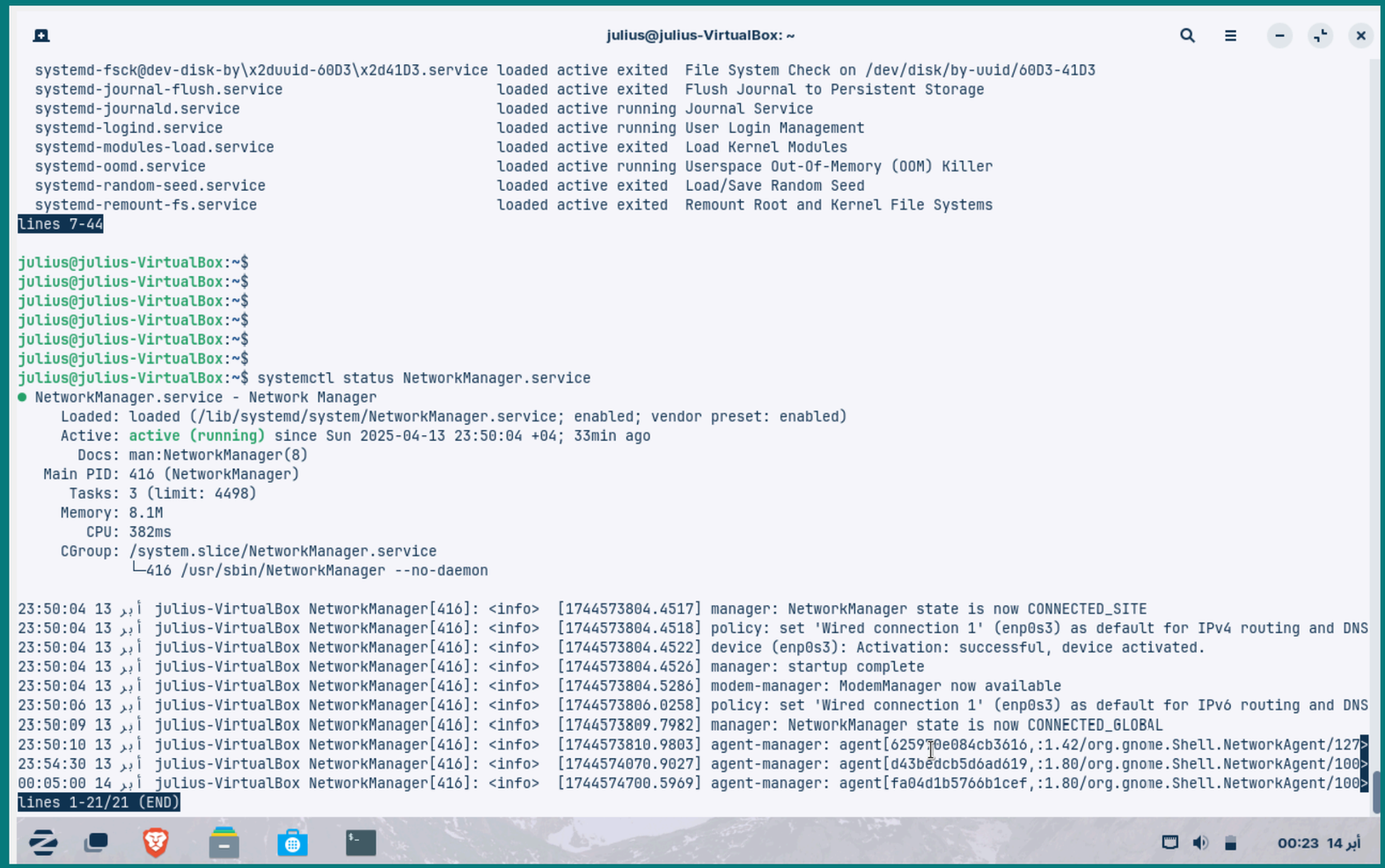
Shows a tree view of all processes

Viewing Daemons:systemctl list-units --type=service}



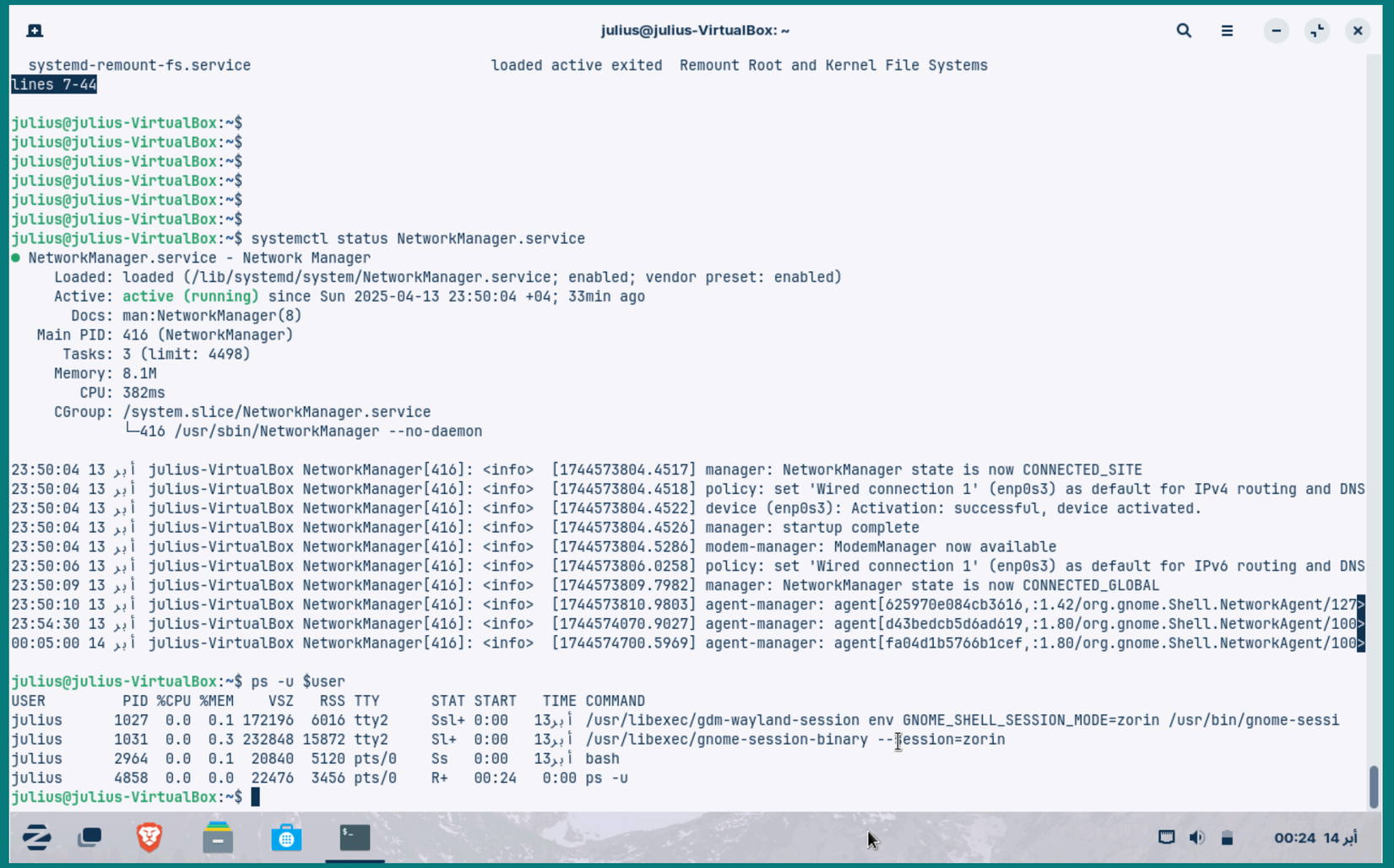
Lists all the system services and their statuses

Specific Daemon:systemctl status NetworkManager.service



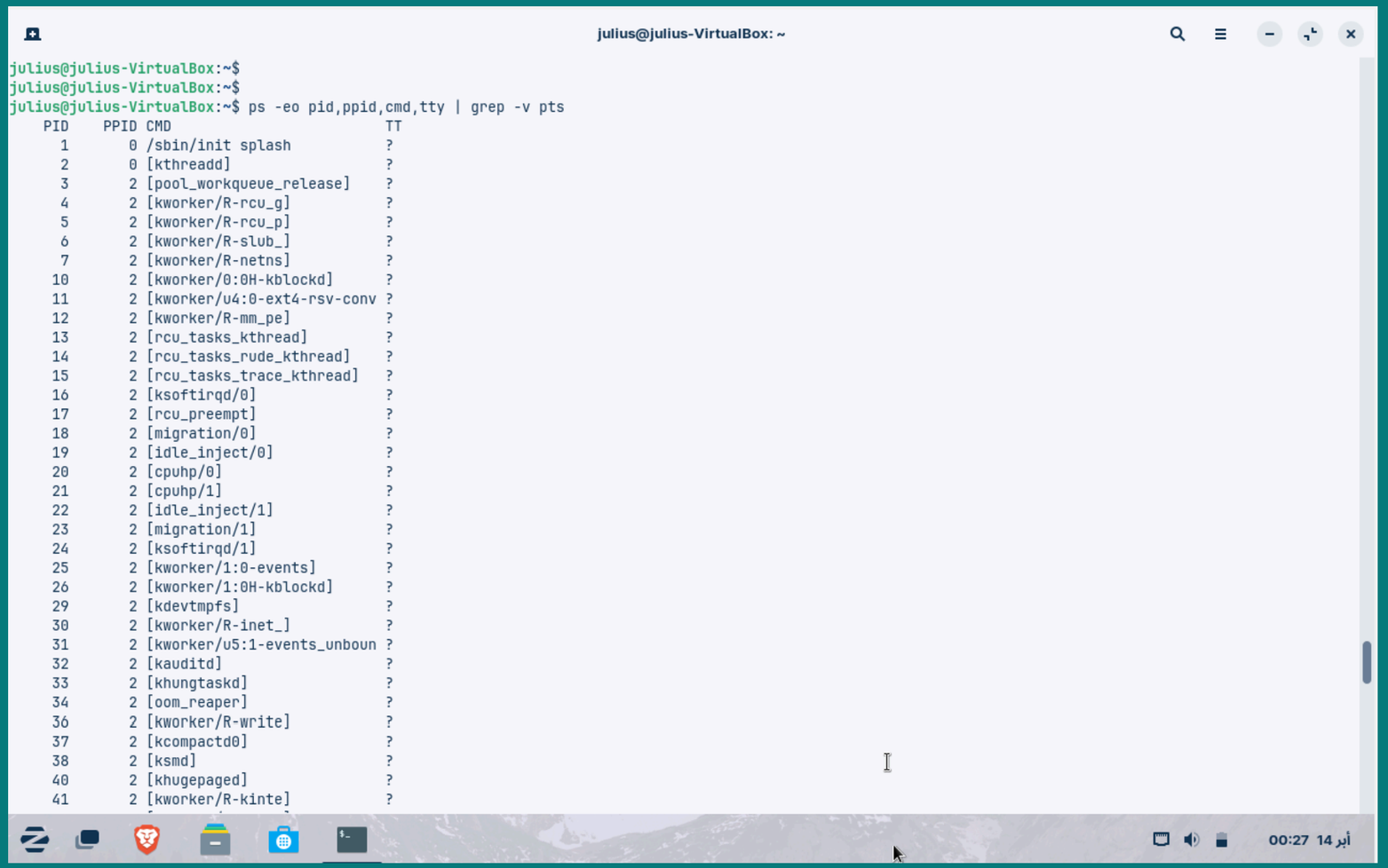
Checks background daemons not shown in GUI

Checks Processes for Current User: ps -u $USER



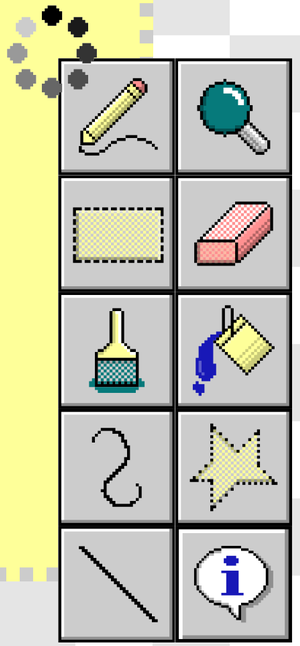
Shows only the processes running under your current user julius

View Active Daemons:ps -eo pid,ppid,cmd,tty | grep -v pts



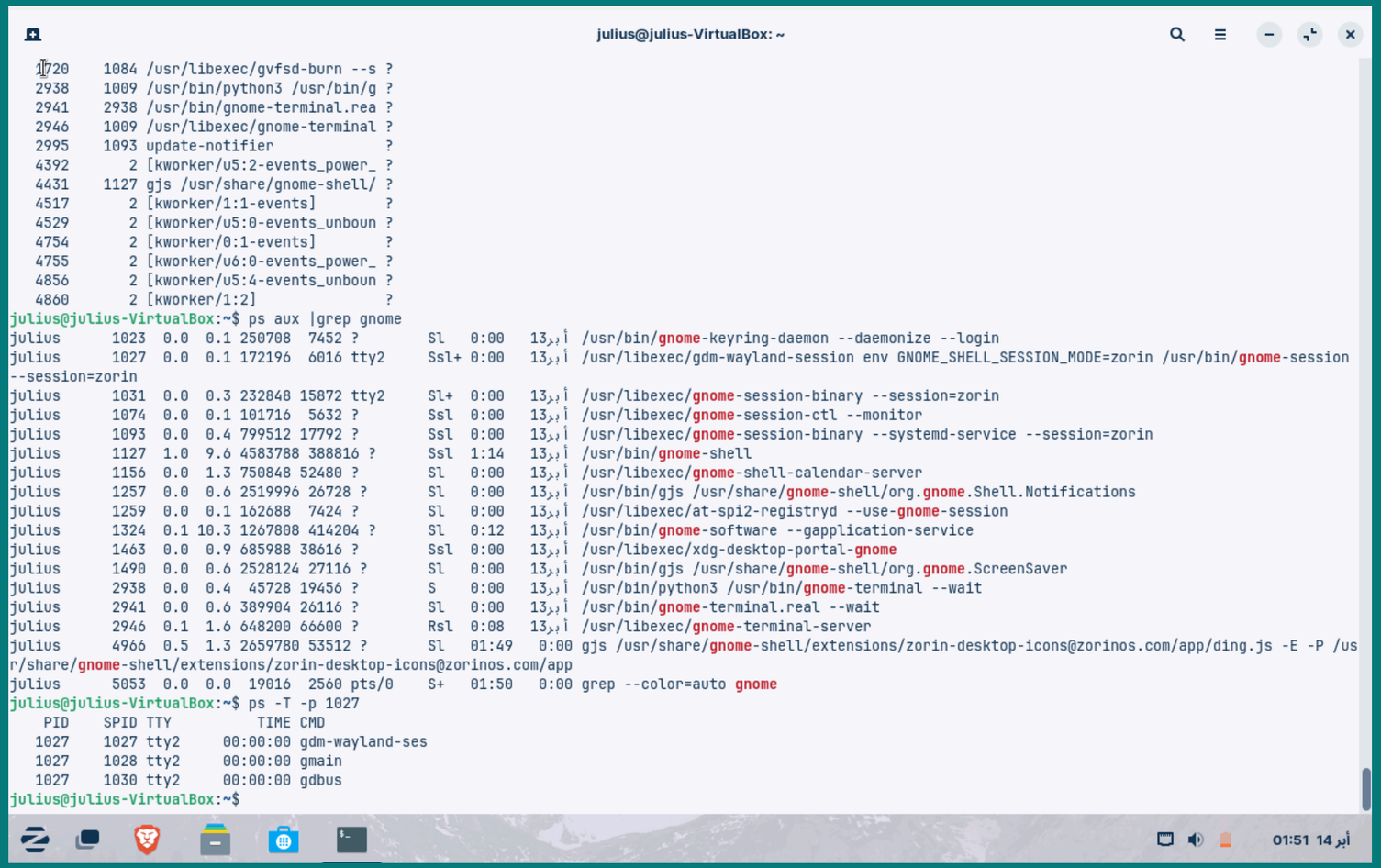
Filters out user terminal processes and shows only the Daemons

Terminal Implementation of Threads



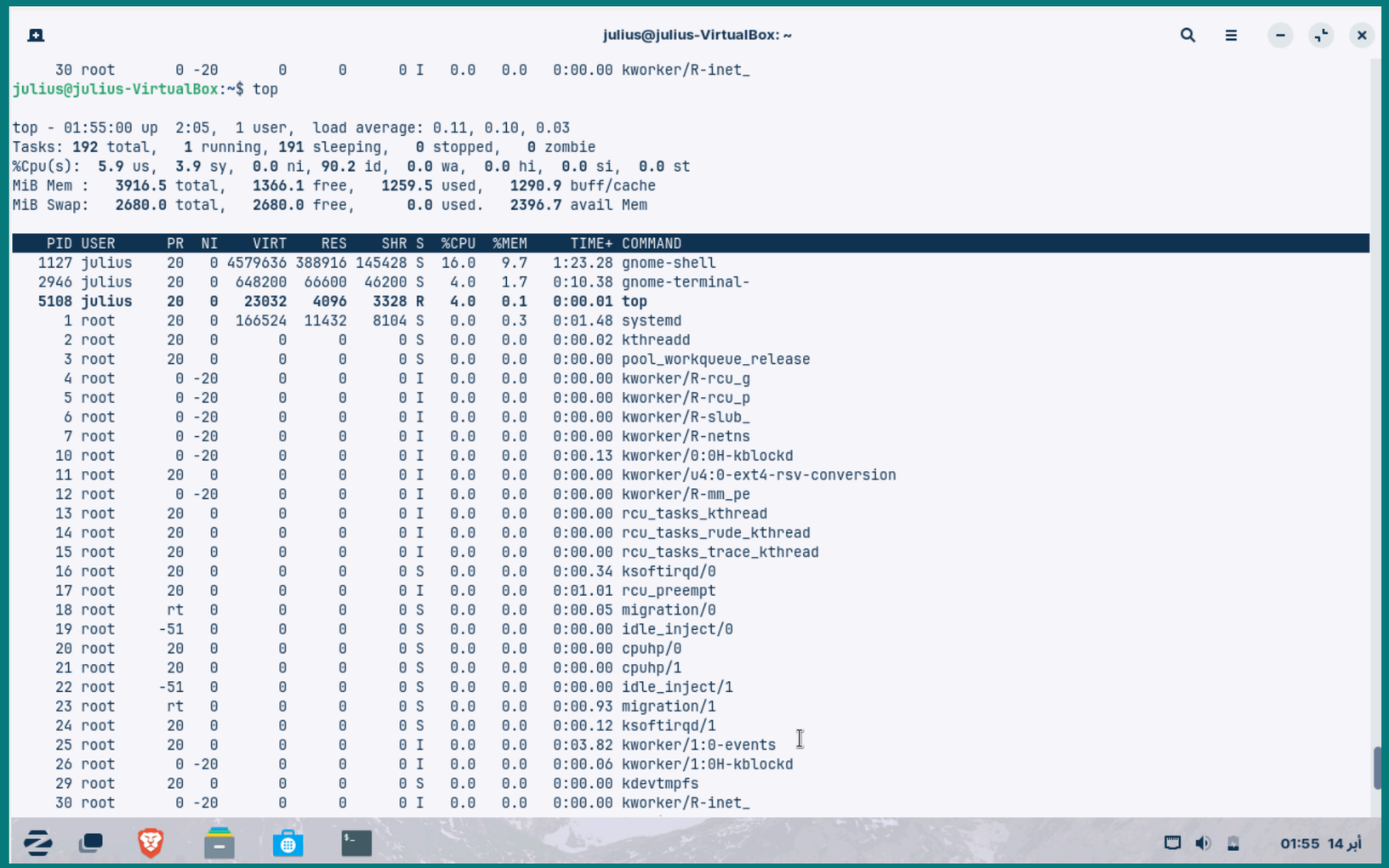
Back to Agenda Page

Find the PID (Process ID):ps aux | grep gnome then find the Thread using ps:ps -T -p <PID>



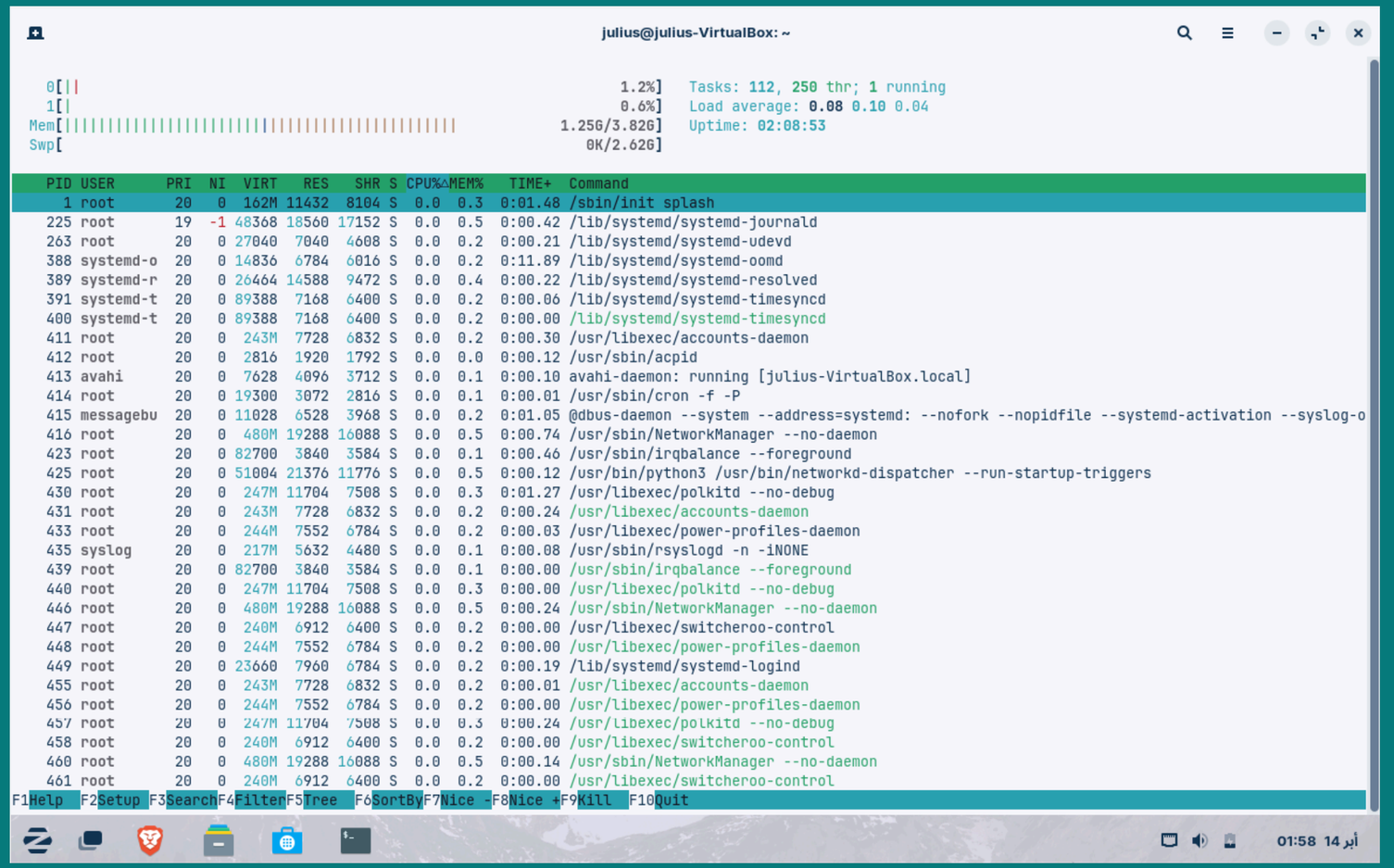
This will list all threads known as LWPs (Light Weight Processes) under that process

Using top to View Threads, start with top then press H



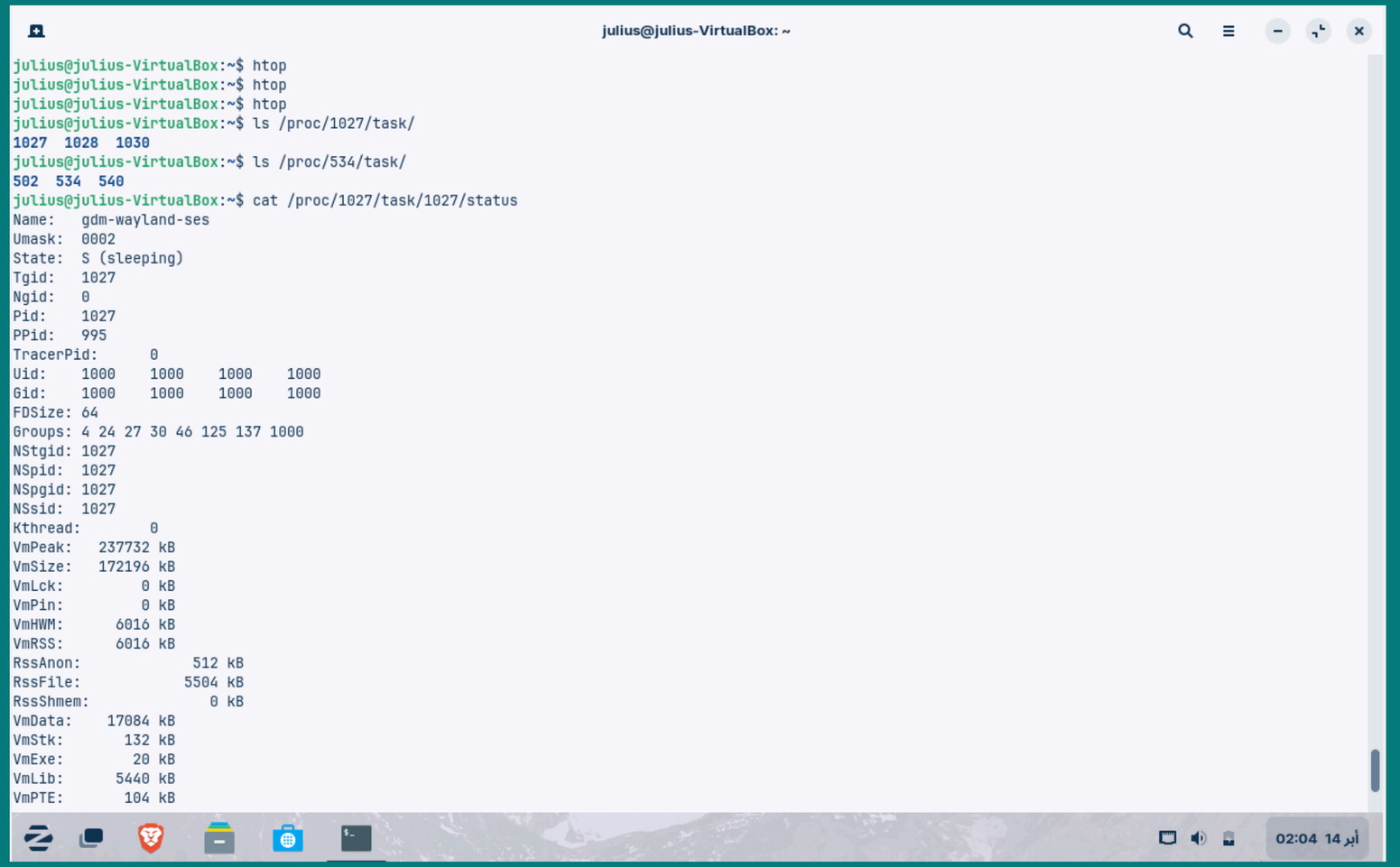
By pressing H we can change from showing processes to threads, so each thread will have its own LWP ID.

Use of htop for Thread view. Begin with htop then press H



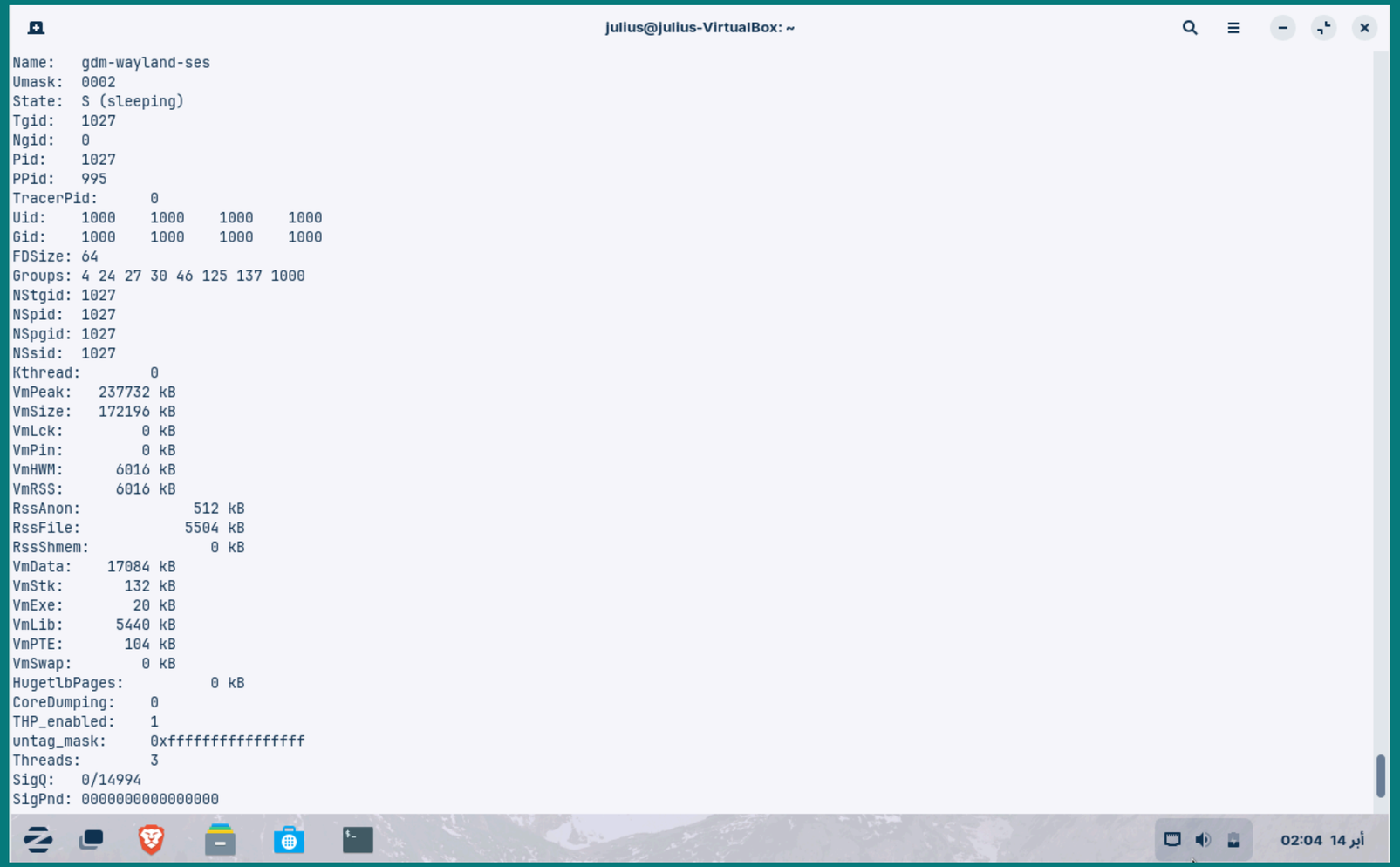
Shows threads indented under parent process with LWP numbers and different CPU usage per thread

Deep Dive of File System:ls /proc/1234/task/



For a particular PID each folder inside /proc/1027/task/ and /proc/534/task/ corresponds to a thread ID (TID) of that process

An even Deeper Deep Dive:cat /proc/1027/task/1027/status



Shows the status of the main thread. Using /proc, we can do manual inspection through Linux displays threads and processes.

Thread Analysis

Through these screenshots, we can get a detailed overview of the thread behavior in Zorin OS. The primary tool used here was terminal, more of cmd version of Windows OS. The primary commands used as written above are top, htop, /proc filesystem inspection and ps shows us how the threads are handled in the user and kernel space. The system showcases a typical Linux thread architecture where Light Weight Processes (LWP) or threads share significant resources like memory and PID namespaces simultaneously monitoring different states of running. One important takeaway here is from /proc/1027/status which refers to the process gdm-wayland-ses. Generally, this process was randomly chosen to represent in screenshots and analyze. It is responsible for dealing with the GNOME display manager, and spawns three threads (IDs 1027, 1028 and 1030). As indicated in the image, these threads are in a sleeping state S which means they are idle and hence waiting for events like user input or system calls. Moreover, through these memory metrics, we can see the shared memory usage like VmRSS for resident memory and even RssAnon for thread-private allocations which signifies that threads in a same process share resources but have separate stacks and execution contexts.

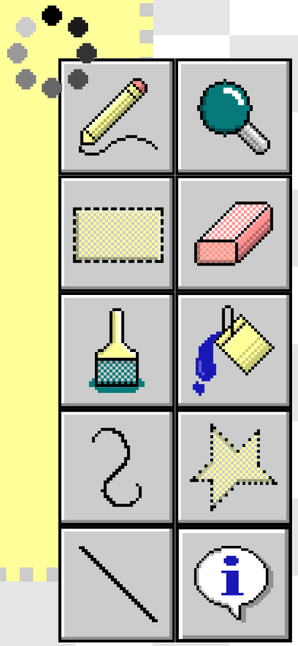
Thread Distribution

The outputs resulting from the command top allow us to view from a wider perspective. We can observe that the system reports between 192 to 443 total threads with most of them in a sleeping state. One of those threads however, is running actively when the screenshot was taken, representing low system load. With high resource consumers such as PID 1127 (gnome-shell) and PID 2946 gnome-terminal are at the highest in user-space threading while threads related to kernel like kworker, rcu are used the least in terms of resource utilization but are crucial for operations in systems. The difference in total thread counts in the images accounts to the dynamic nature of thread creation and destruction, especially in GNOME which is GUI environment.

Kernel VS User Threads

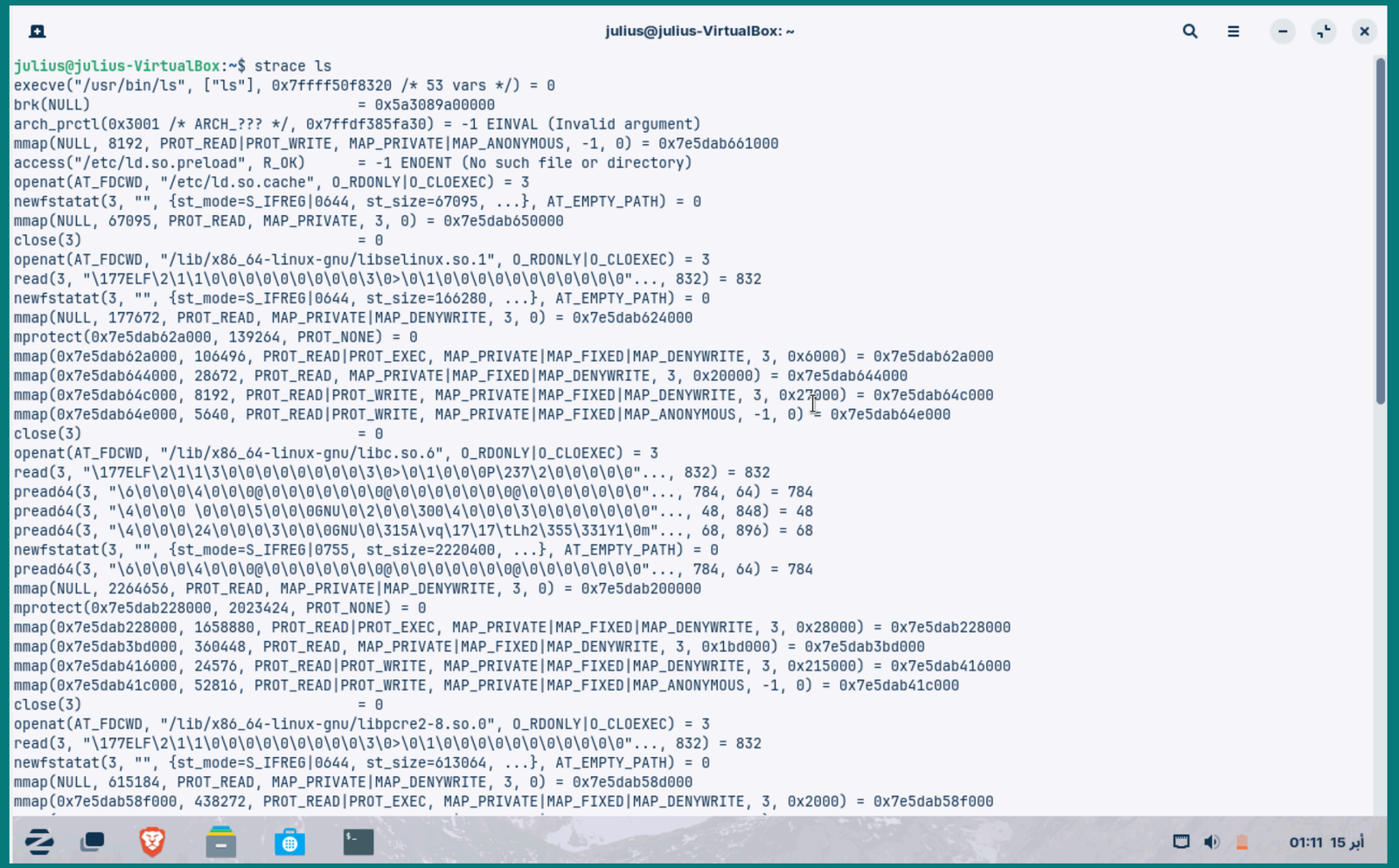
In the results produced by the command top, the Kernel threads as [kworker] or [rcu] are running with the least amount of overheads with near-zero memory and CPU usage but are critical for tasks such as interrupt handling and memory management. On the other hand, user threads, particularly which come under gnome-shell exhibit higher resource demands. We can observe that threaded applications like GNOME terminal gnome-terminal-and systemd services systemd-udevd leverage threading to allow for parallelism. Through Python interpreters /usr/bin/python3 in the process list, we also see the multithreaded scripting, which is common in modern Linux distributions.

Terminal Implementation of System Calls



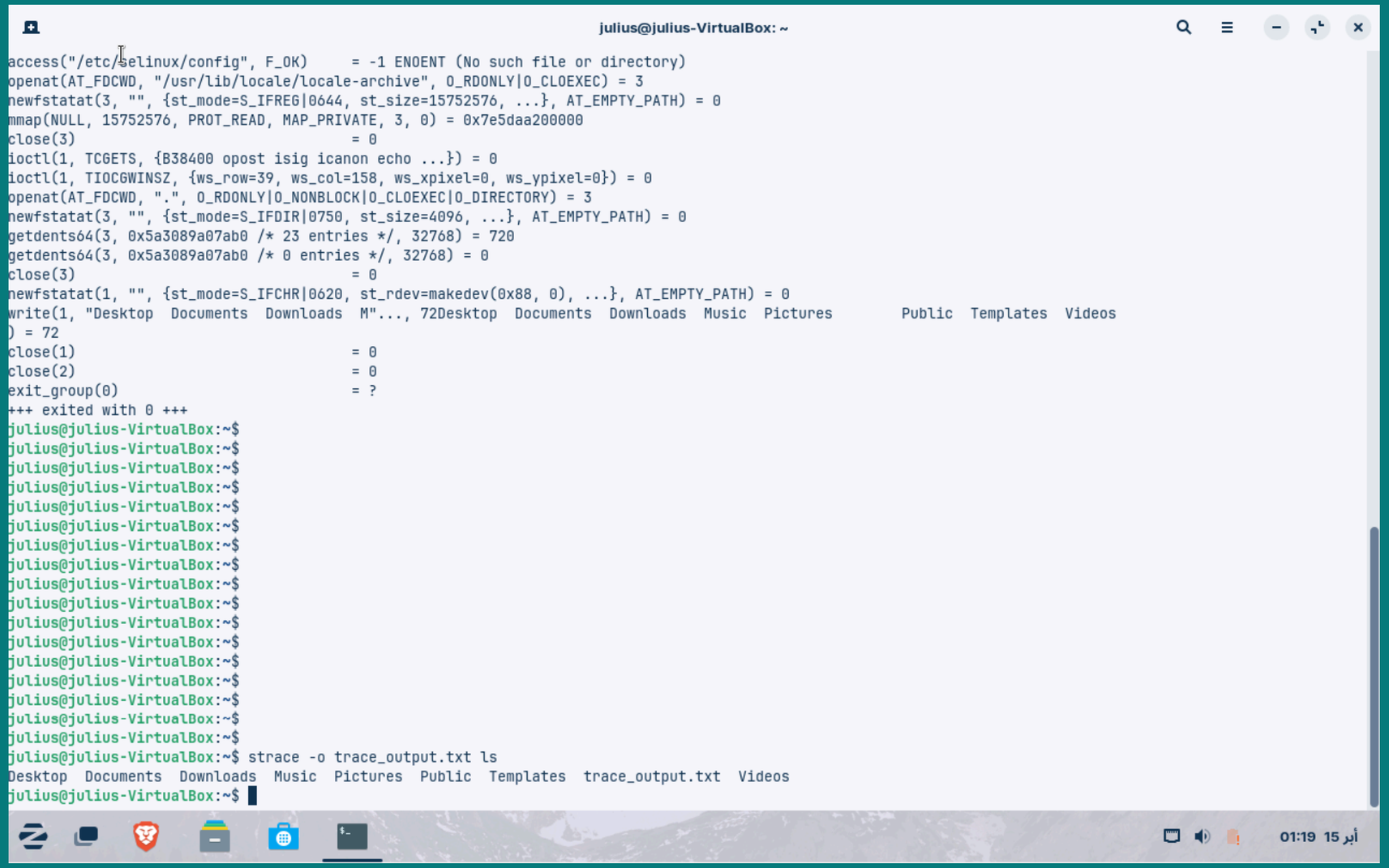
Back to Agenda Page

Using strace: strace ls

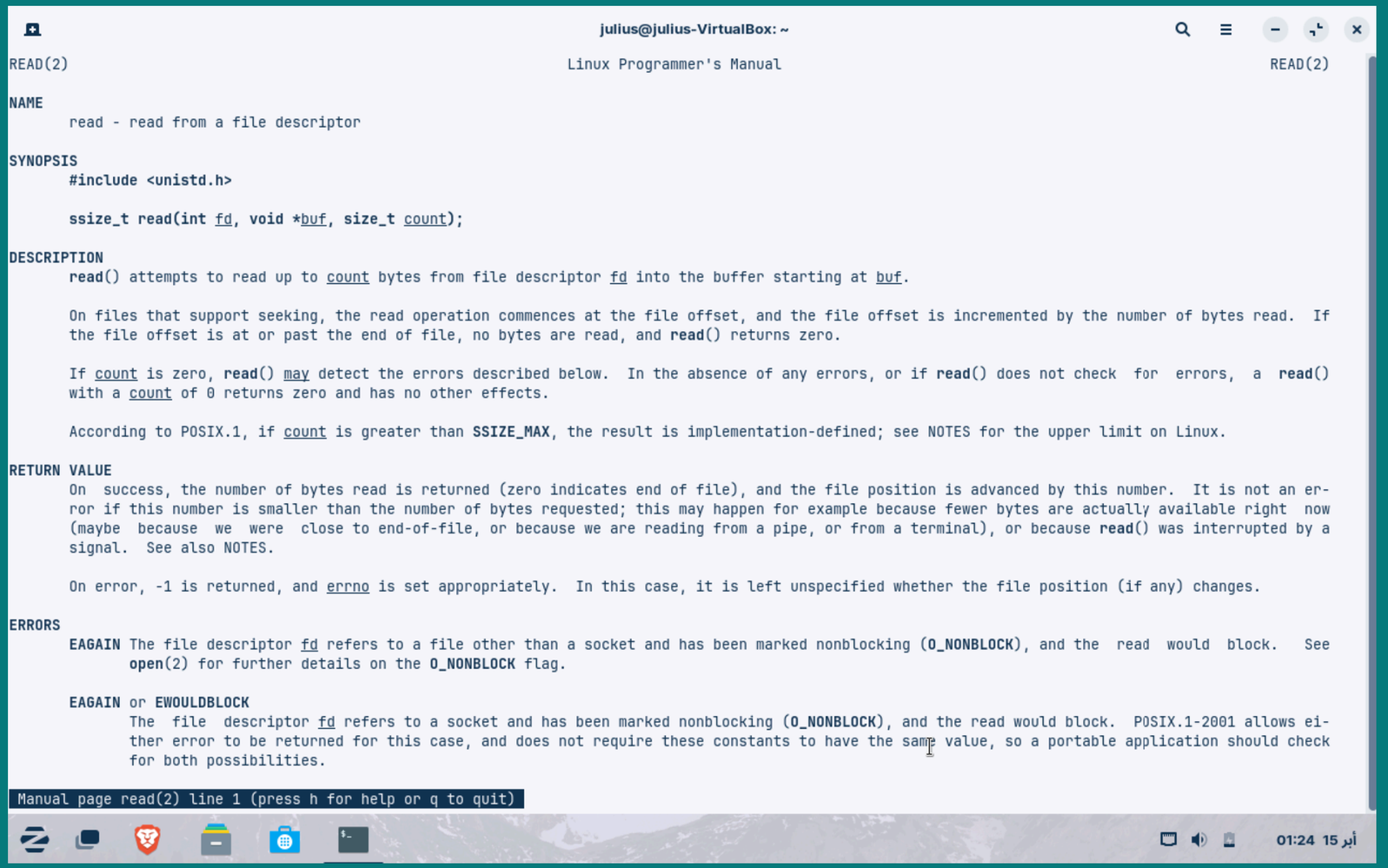


Trace system calls for the ls command using strace ls

Redirecting output to a file using strace -o trace_output.txt ls

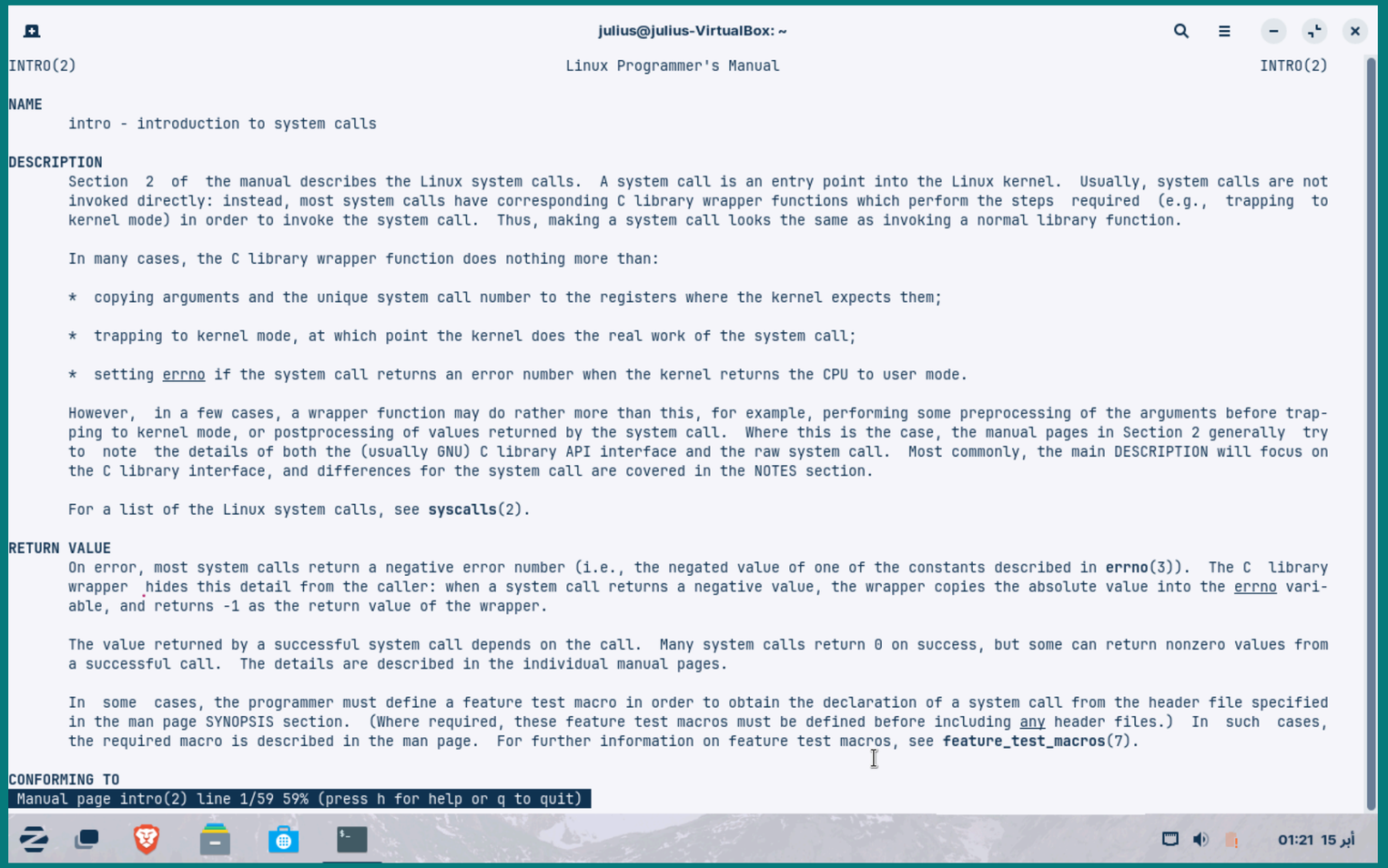


Use of man to see available system calls



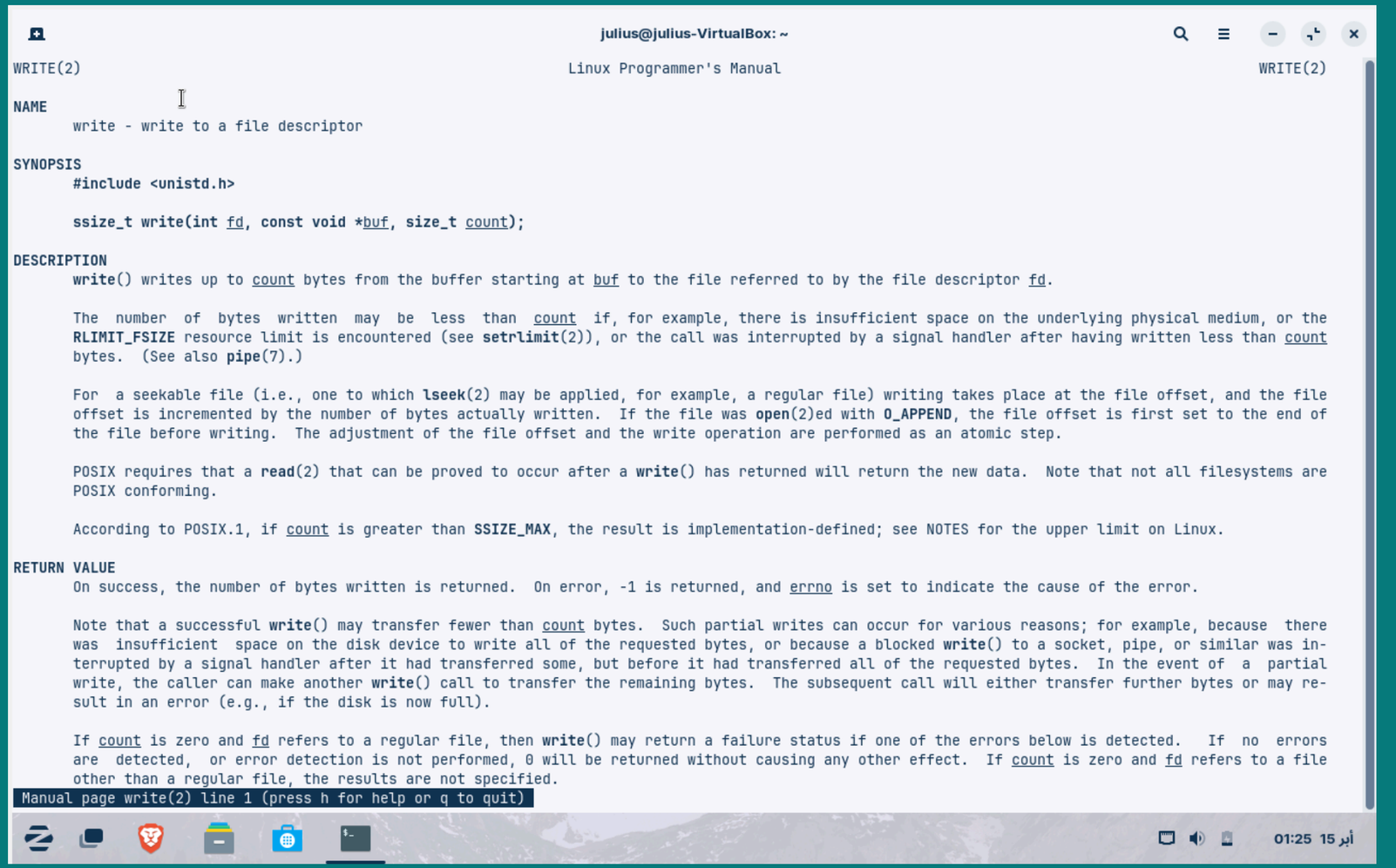
man 2 read shows the manual page for the read system call.

Use of man to see available system calls



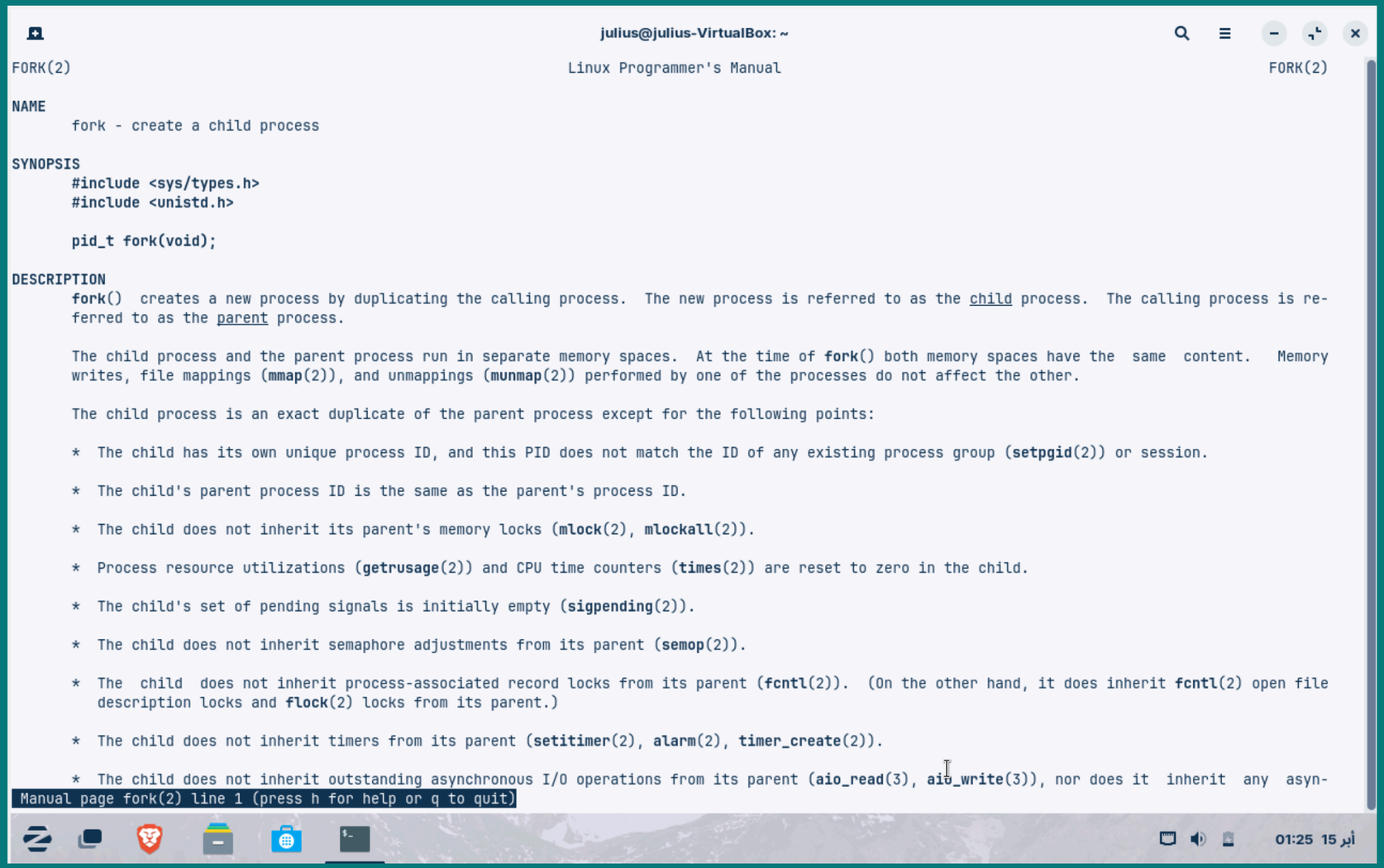
man 2 intro shows an introduction to section 2 of the manual, which is dedicated to system calls.

Use of man to see available system calls



man 2 write shows the manual page for the write system call.

Use of man to see available system calls

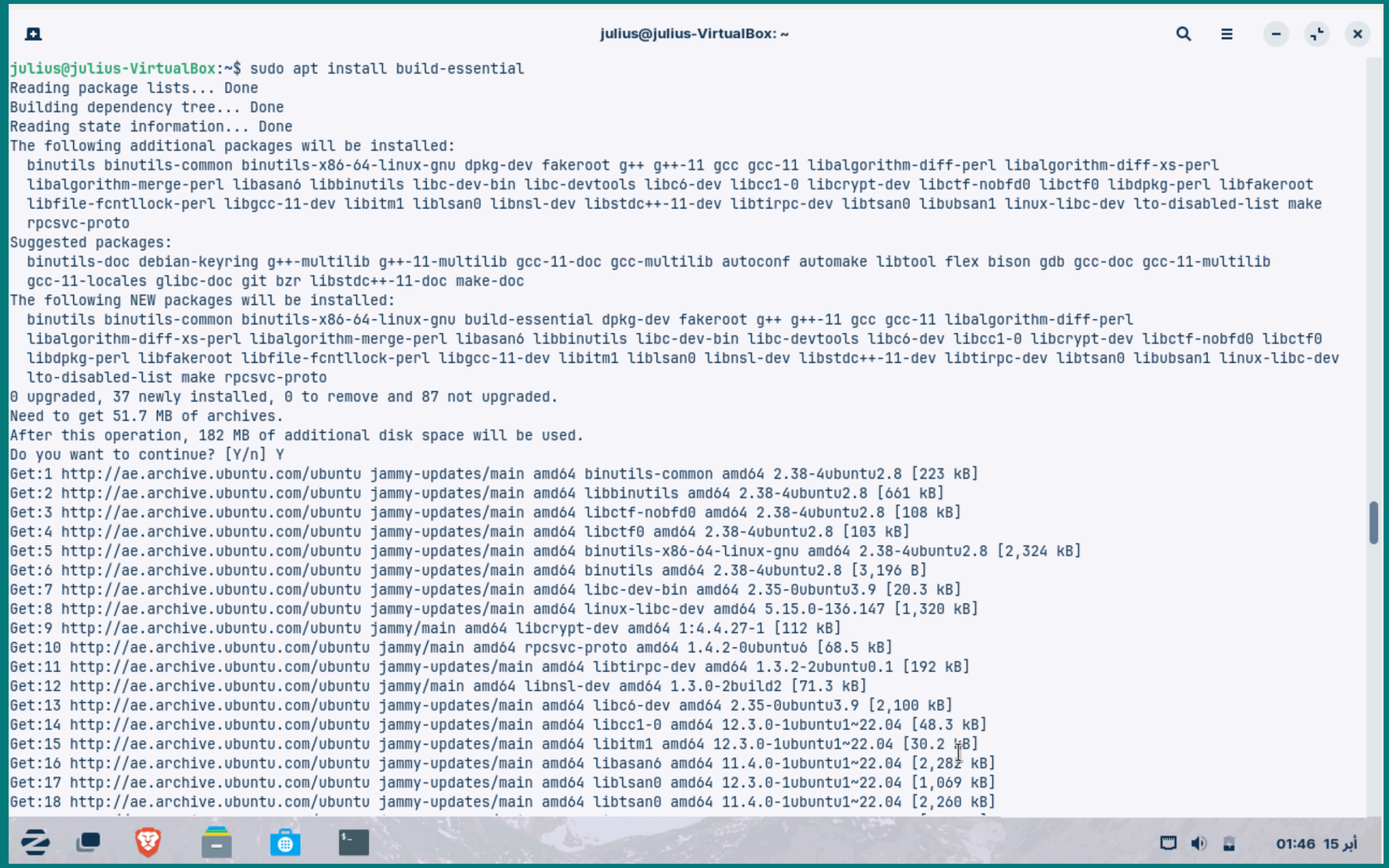


man 2 fork shows the manual page for the fork system call.

ltrace ls

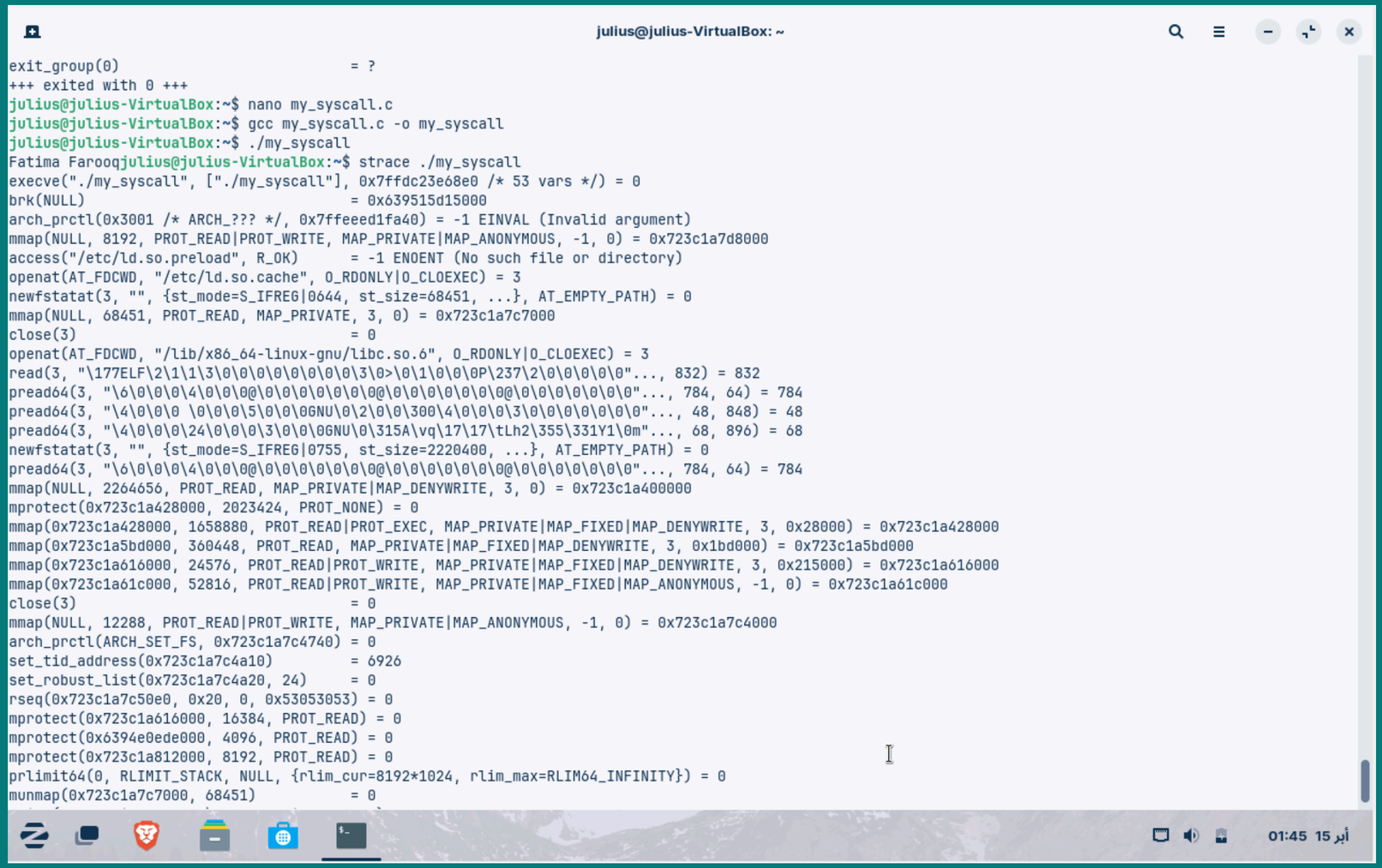
Use of ltrace for library calls (different to system calls)

System Calls using C



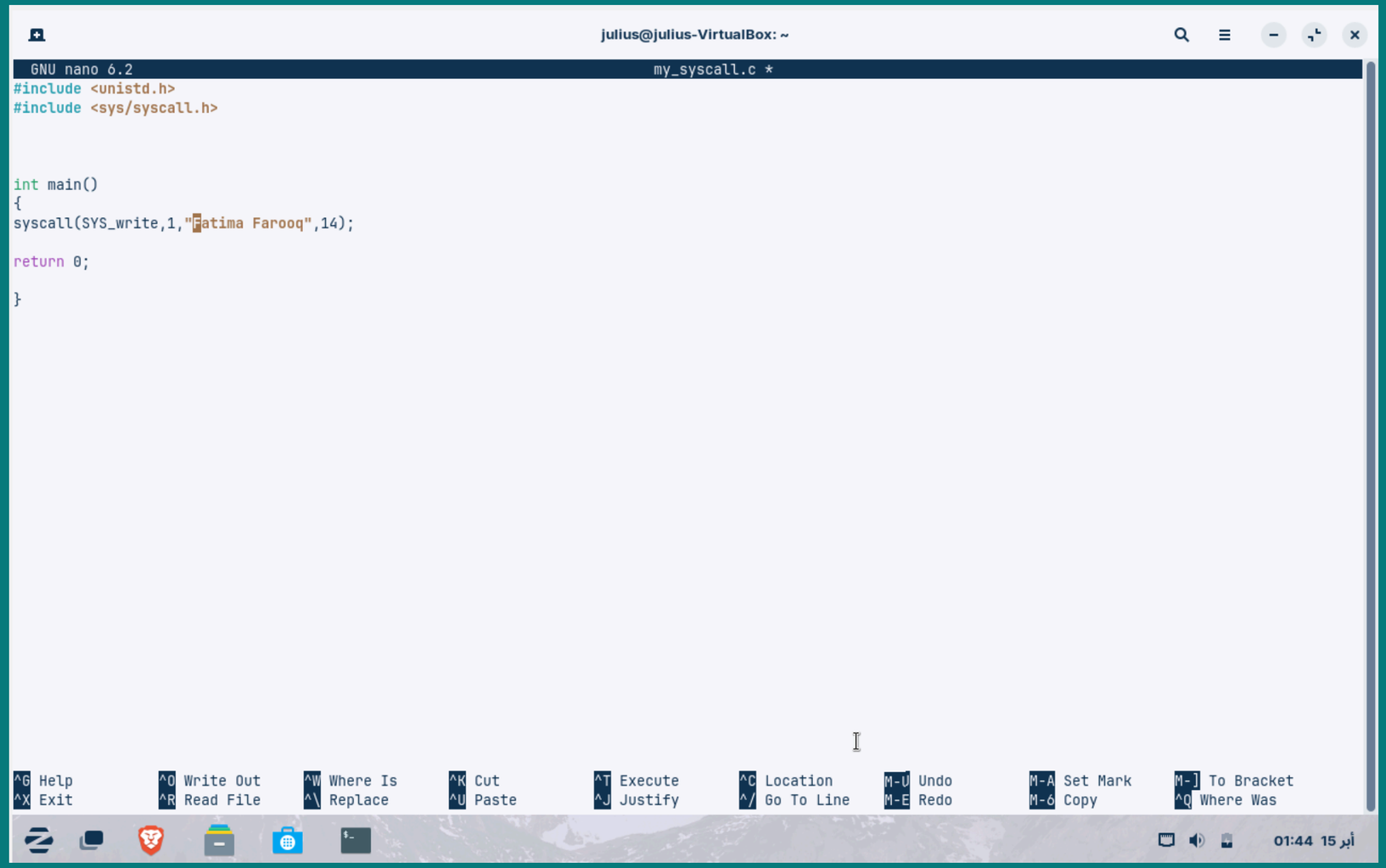
sudo apt install build-essential

System Calls using C



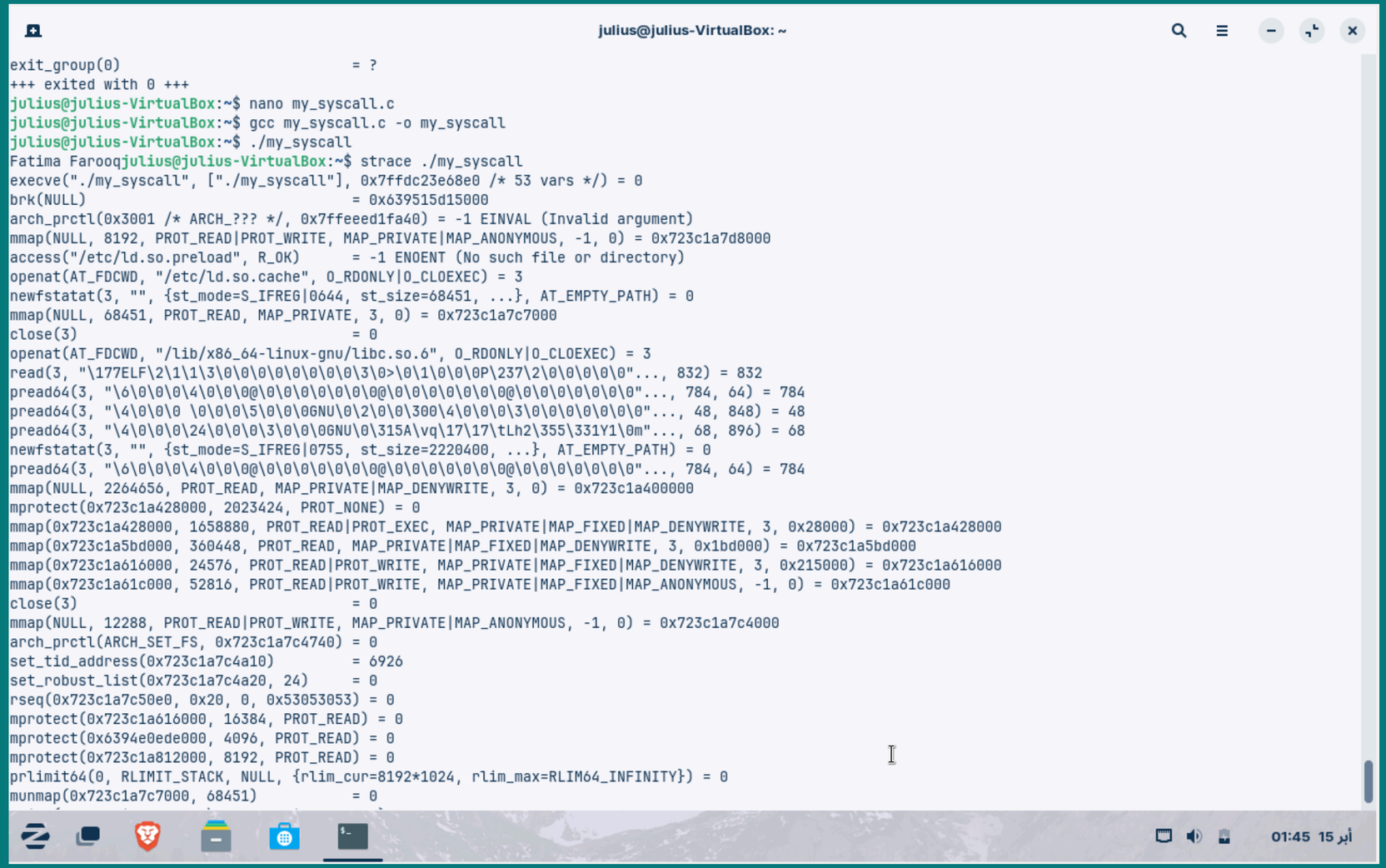
nano my_syscall.c

System Calls using C



Type a simple C code and include the relevant libraries

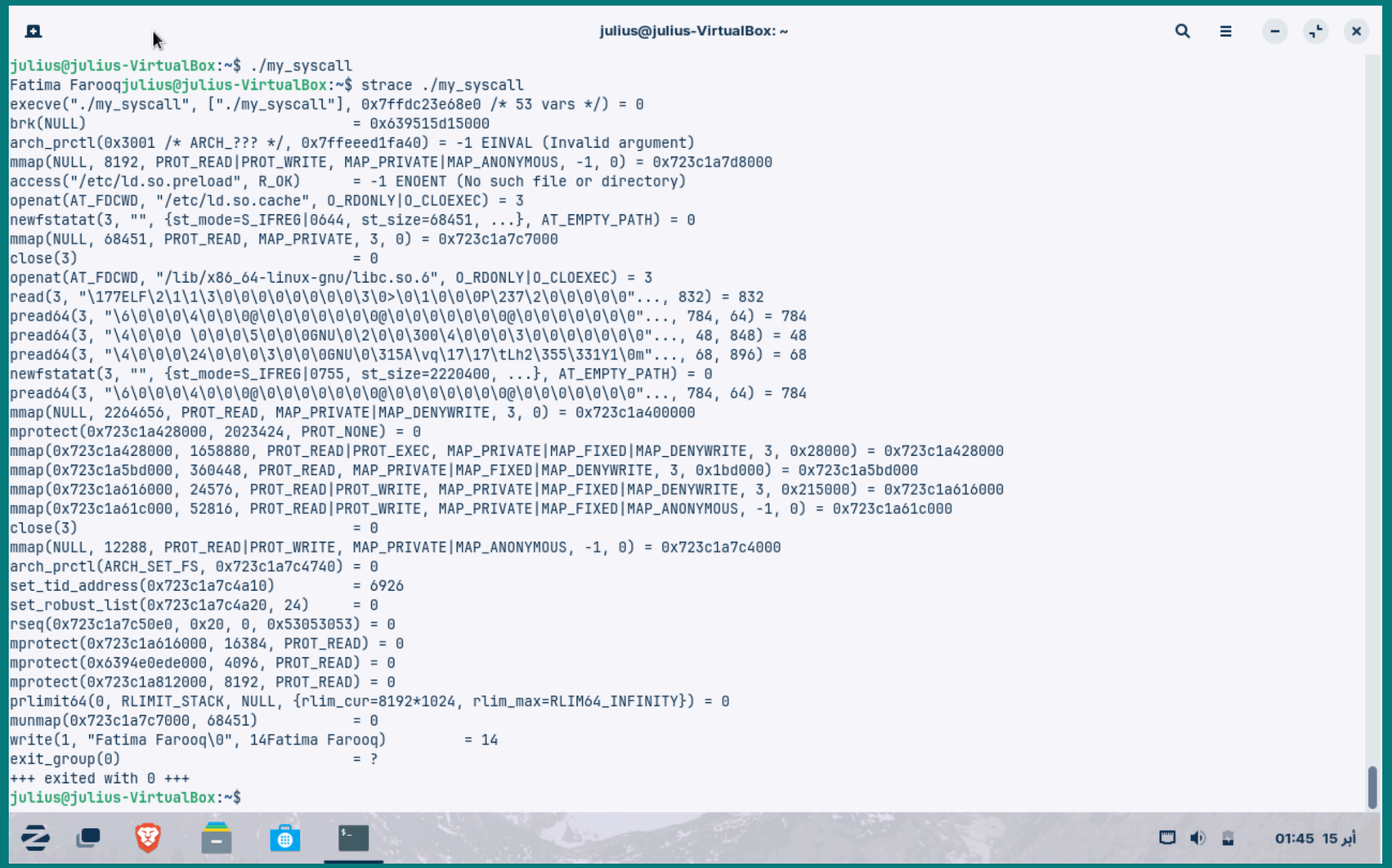
System Calls using C



gcc my_syscall.c -o my_syscall

./my_syscall

System Calls using C



strace ./my_syscall


Thank you!