PREDATOR-OS

SECURITY, PRIVACY, HARDENED, ANONYMIZED

Penetration Testing and Ethical Hacking

Polymorphic Security Platform

Developer: Hossein Seilani

1300 pre-installed tools which are split into 40 several categories

Predator-OS Linux
User Guide

Polymorphic Security Platform

A security-centric free open-source Linux

Penetration testing and Ethical hacking and can be use it as: privacy, hardened, secure, anonymized

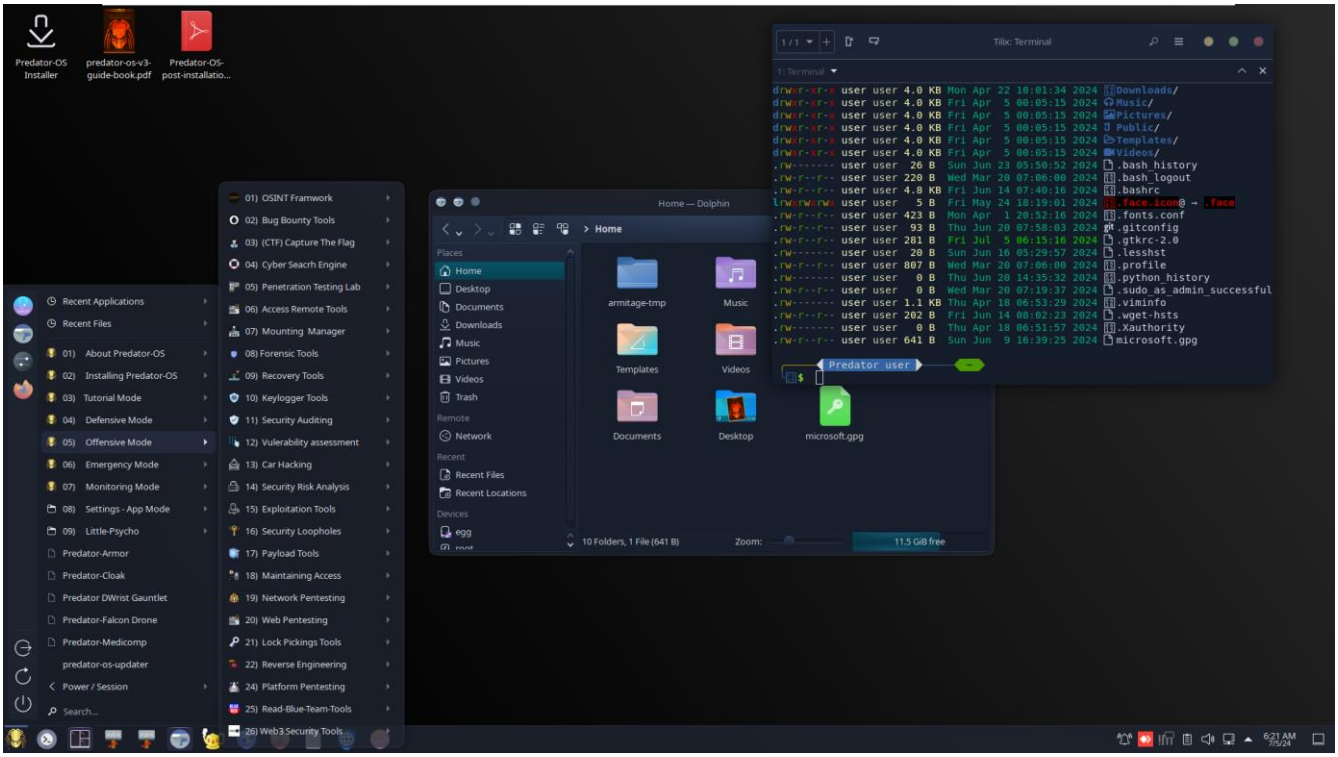
By **hossein seilani**

(2024)

Preface

It was developed in 2021, by **Hossein Seilani** who is also the developer of **https://emperor-os.ir/** Linux too. The Predator-OS is a free open-source community project, Free (as in freedom). The distro is for penetration testing and ethical hacking and also privacy, hardened, secure, anonymized Linux. Predator Linux is based on Debian, kernel 6.6 LTS and 6.1 LTS and using a fully customized plasma desktop with a special menu of tools.

Predator Linux has around 1300 pre-installed tools, which are split into 40 several categories. Predator Tools are imported from both Debian and Debian stable repositories and GitHub page. Most kernel and user configs are customized by default to prevent any hacking, non-privileged access and reduce the attack surface. Many built-in firewalls and defensive tools allow end-users to control the Predator-OS. Predator also supports much privacy, anonymized, security tools, and also both it to be run as Live-CD or from a USB Drive and installation mode.



Details

- OS Type: Linux
- Based on: Debian Stable
- Kernel: 6.6
- Origin: Emperor-os Team , Iran
- Desktop: Plasma
- Other Desktop: as soon as possible
- Category: penetration testing, security, privacy, Forensics, Live Medium, hardened, anonymized

Downloading a Predator-OS ISO Image

Where to Download

https://www.seilany.ir/predator-os/download/Predator-OS-v3.1-amd64-26-06-2024.iso

Distribution	Predator-OS
Home Page	https://Predator-OS.ir
Mailing Lists	Info.predator.os@gmail.com
User Forums	**http://t.me/predator_os**
Documentation	https://Predator-OS.ir

Why this book?

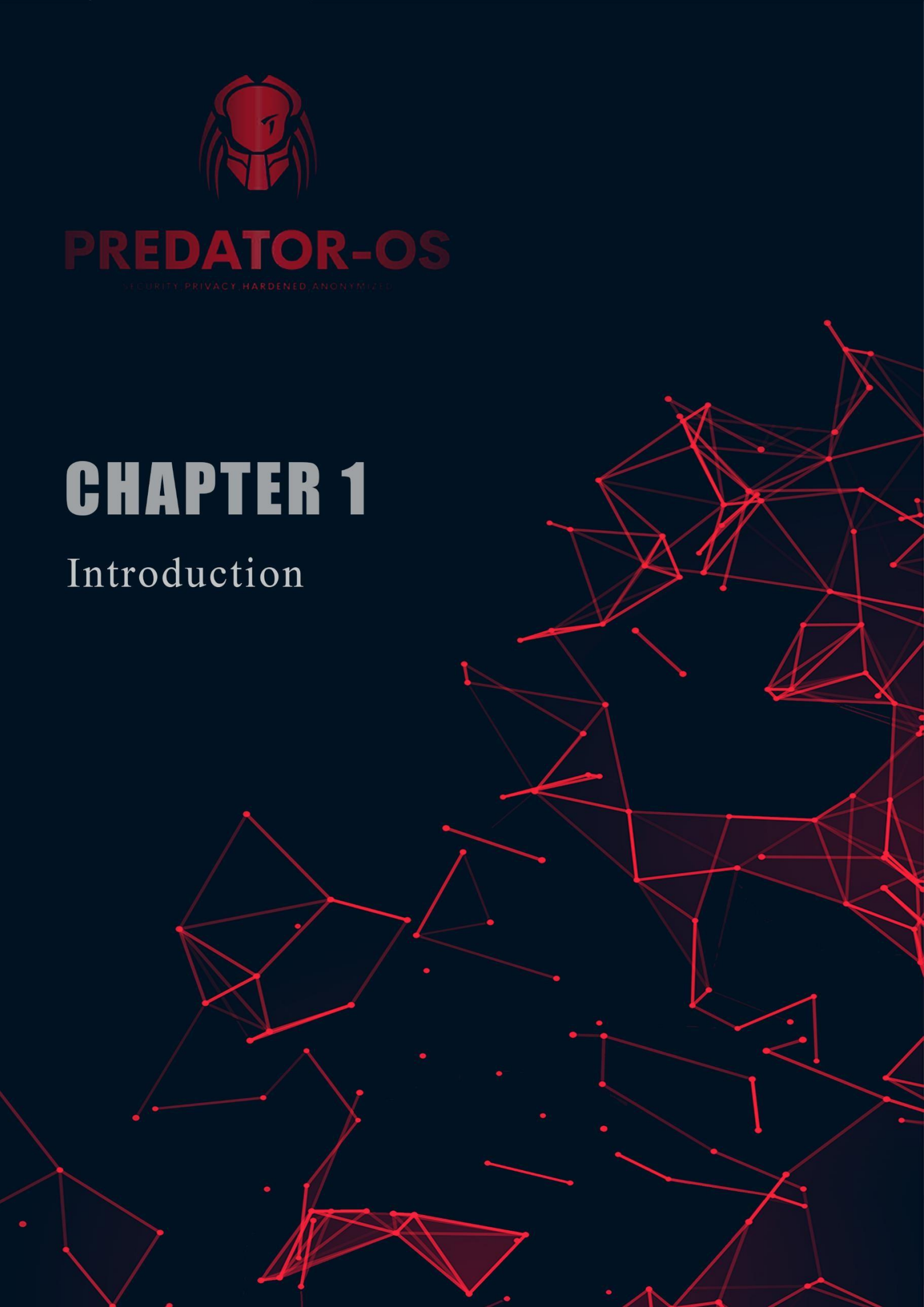
Predator-OS Linux is not built to be a simple set of tools, but rather a flexible, Polymorphic security platform that professional penetration testers, security enthusiasts, students, and amateurs can customize to suit their specific needs. Also, the Linux Predator-OS is not just a collection of various information security tools pre-configured to prepare you. To get the most out of Predator, it's important to have a solid understanding of Linux and how to use them in your environment.

Although, Predator-OS is multipurpose and works in 10 different security modes. But it is primarily designed to help with penetration testing. Also, this book is not only to help you when using Linux Predator-OS, but also to improve your understanding and simplify your experience so that when you are involved in a penetration test. You don't have to worry about losing precious minutes to install new software or new settings or activate a new network service. In this book, first you will be introduced to Linux, then you will get to know Predator-OS in more detail.

This book will help to better understand this operating system and is intended to help both beginners and professional Linux users, as well as users who are looking to deepen their knowledge about security settings. In addition, this book can be used as a road map, or a training reference on Linux operating system configuration and security settings.

This book is designed so that you can focus on Linux Predator right from the start.

Contents



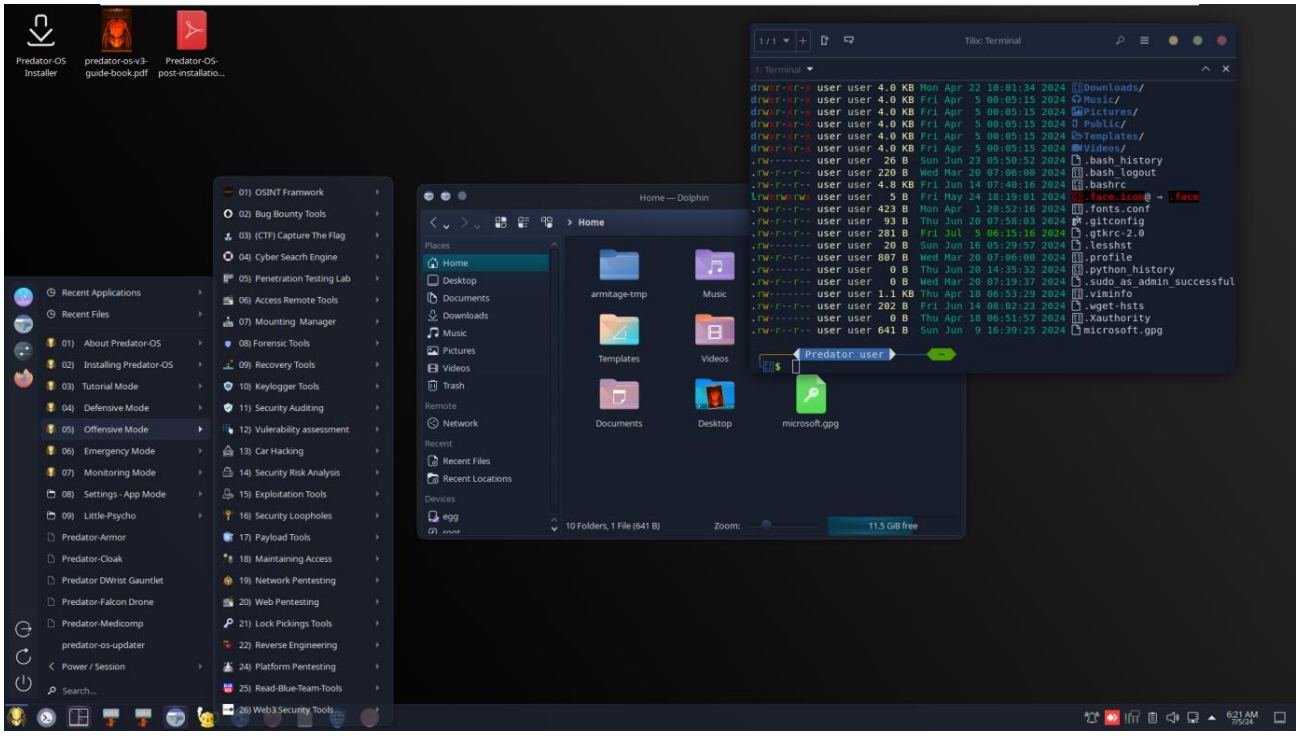
CHAPTER 1

Introduction

Chapter 1

What is Predator-OS?

Predator is a GNU/Linux distribution. It is based on Debian stable. It is a complete operating system for Cybersecurity users and environments. Including software and systems for installation and management all based on the Linux kernel and free software. When I created Predator-OS, in 2021, I sought to have two principal features. First, performance and more security. It would also be a non-commercial distribution. I focused on the predator to be a **polymorphic security platform** for beginners and professional users and for academics and universities.



Remember that hackers are big people. Those who create valuable things and build the world. On the other side, there are crackers who only lose money. We have a lot of respect for real hackers. Those who usually hide in the shadows and whose names are rarely heard.

Predator-OS is a good choice for those who love security. This distribution has a great advantage over its competitors and has an easy and attractive user interface. Even if you are not familiar with Linux, you will feel good about this distribution. He is well-versed in the philosophy of open source and has applied this point in his development process. There are many open-source tools in this operating system for database analysis. Wireshark provides network packet analysis tools for analyzing information in networks, Bluetooth, wireless networks, databases, forensics, and more. Predator has been in development since 2021. This operating system supports a 64-bit platform. You can test Predator before installing it on

live. It uses the PLASMA desktop by default and is released under the GPL general license.

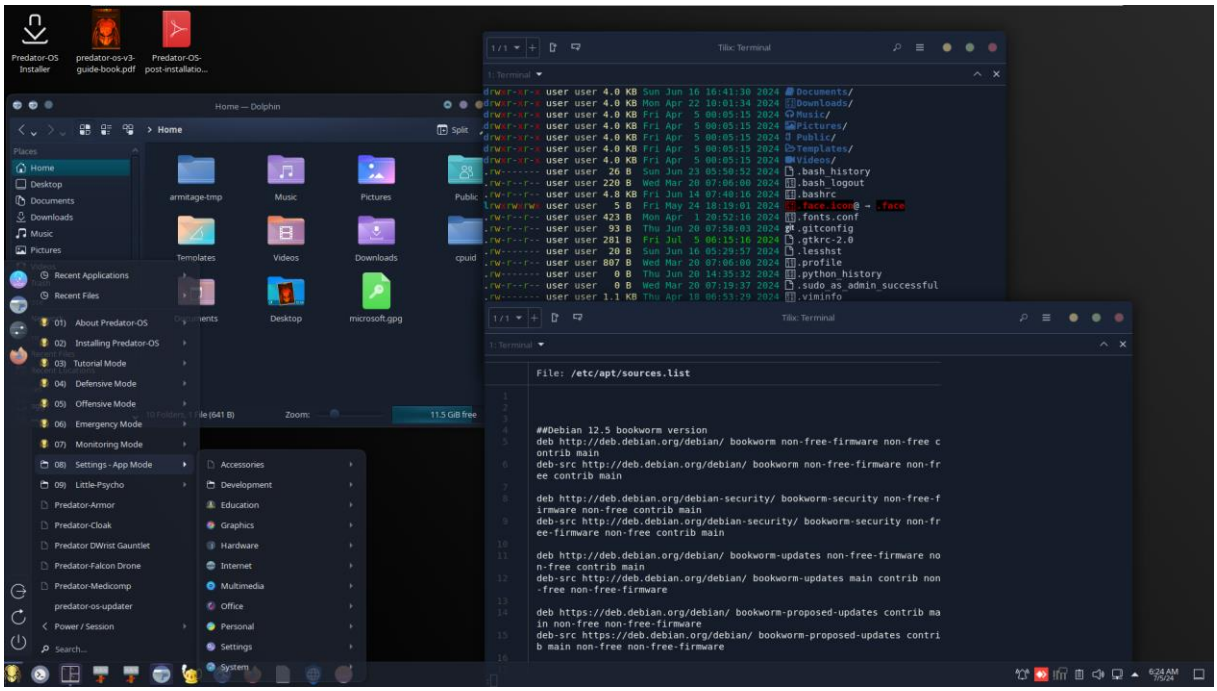
Why Predator-OS Linux?

You install Kali Linux; but after installing it, you realize that it is hardly usable. Despite the advanced Kali kernel, the network cards and mouse don't work well after installation, and the heavy NVIDIA graphics card and GPU lack properly installed drivers. In Kali Live mode, it indicates that these advanced drivers have not reached the core yet.

This is especially true for those who are drawn to the security field, whether it's a hobby, a hobby, or a line of work.

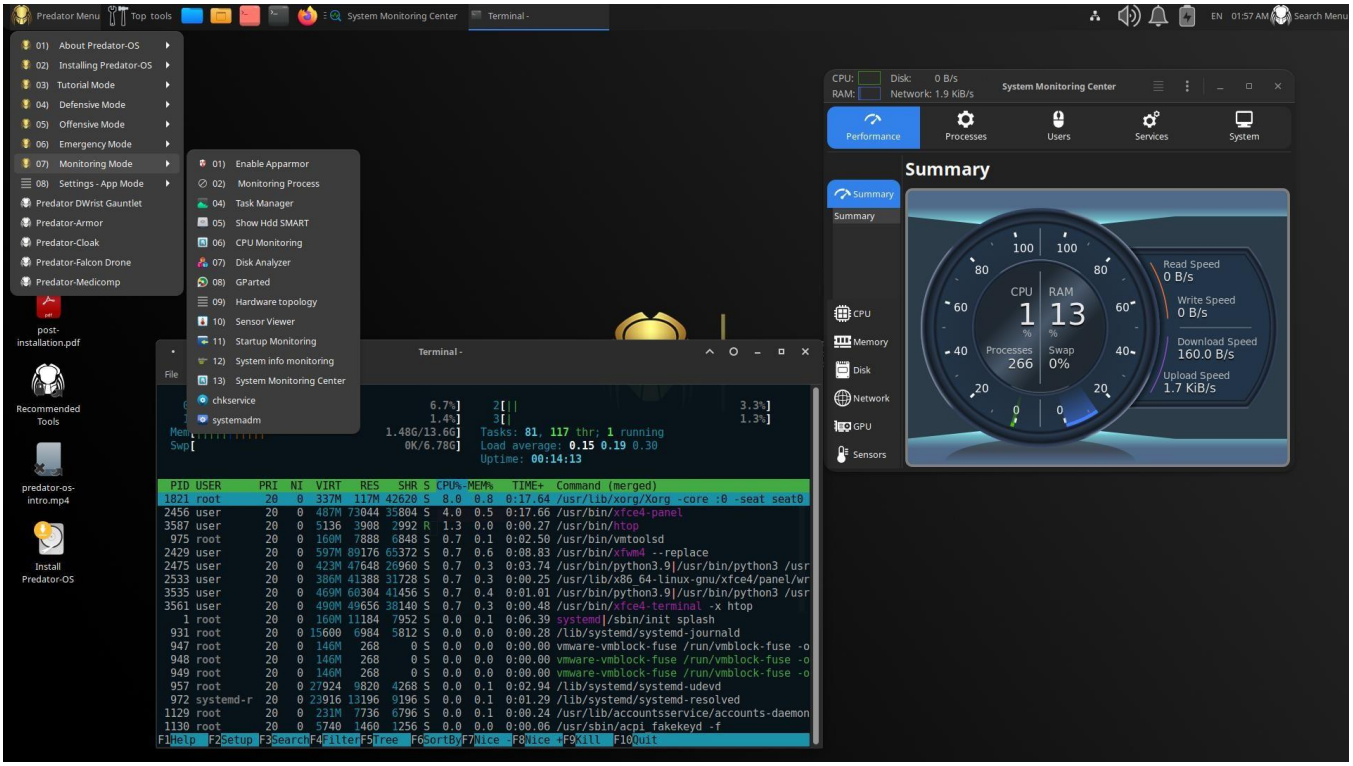
By observing these problems among users, we realized that we could guide users into the world of security by creating a more structured and user-friendly Linux distribution. Helping our community while simplifying all the complexities of Linux.

Predator-OS Linux provide a polymorphic security platform for systems and network administrators, security experts, digital forensics operations and cybersecurity engineers. It was focused on Pentesting, Ethical Hacking, Secure, Privacy, Hardened and Anonymized Linux.



The Predator-OS is available in edition security now. With other editions such as Desktop, IOT, mobile, Virtual machine, Raspberry Pi and Docker being released soon.

The default Desktop is PLASMA but the other Desktops such as KDE plasma, Mate, Gnome will be released soon. The system is designed to be familiar for the security expert and is easy to use for the new entry student; but it does not try to hide its internals, as other generalpurpose distributions try to do.



Why Predator-OS is different?

The Predator-OS Linux has its own unique features, which you can see 100 features on the site, and it also has features compared to security distributions, including:

1) Easy installation and better hardware support than Kali distribution
2) Suitable for newbies users and useful for general work compared to Parrot and Kali distribution
3) Included all Parrot Linux tools
4) Lighter and lower download file size despite the tools More than the Black Arch distribution
5) Ability to boot live and also installation, compared to the deft Linux despite having all the tools of this distribution
6) included the feature of booting in text mode and having CLI tools such as the dracOS distribution that It lacks graphical tools.
7) Covers all Bugtraq Linux tools

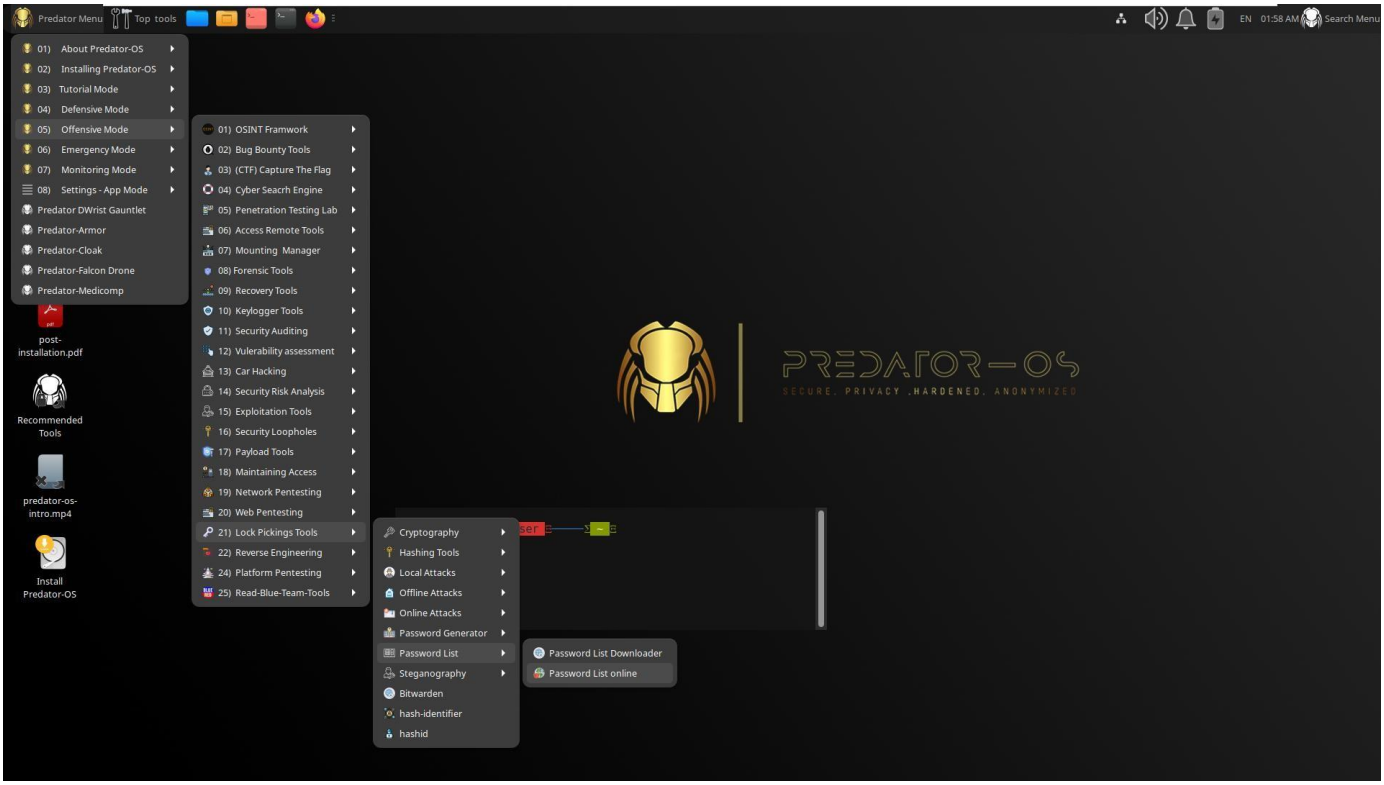
8) Included More web penetration testing tools than Samurai Linux
9) Included More tools than BackBox Linux
10) Included all Pentoo Linux tools
11) Included specialized PC crime detection tools and Also the ability to run Windows tools in Linux, such as deft and CAINE Linux
12) Included Kodachi Linux features in the field of privacy and anonymity
13) Included secure and privacy features of Discreete Linux
14) Included all Santoku Linux tools in the field of Mobile pentesting
15) Included the Whonix distribution features for more security
16) Included all Attifyos Linux tools in the field of IoT penetration testing and even more with user-friendly interface
17) Included all stressLinux tools in the field of stress testing and more
18) Included the Features of anonymity on the web such as IprediaOS distribution
19) Cover many of the tools of the following site: insecure.org

New features Included in Predator-OS v3.1

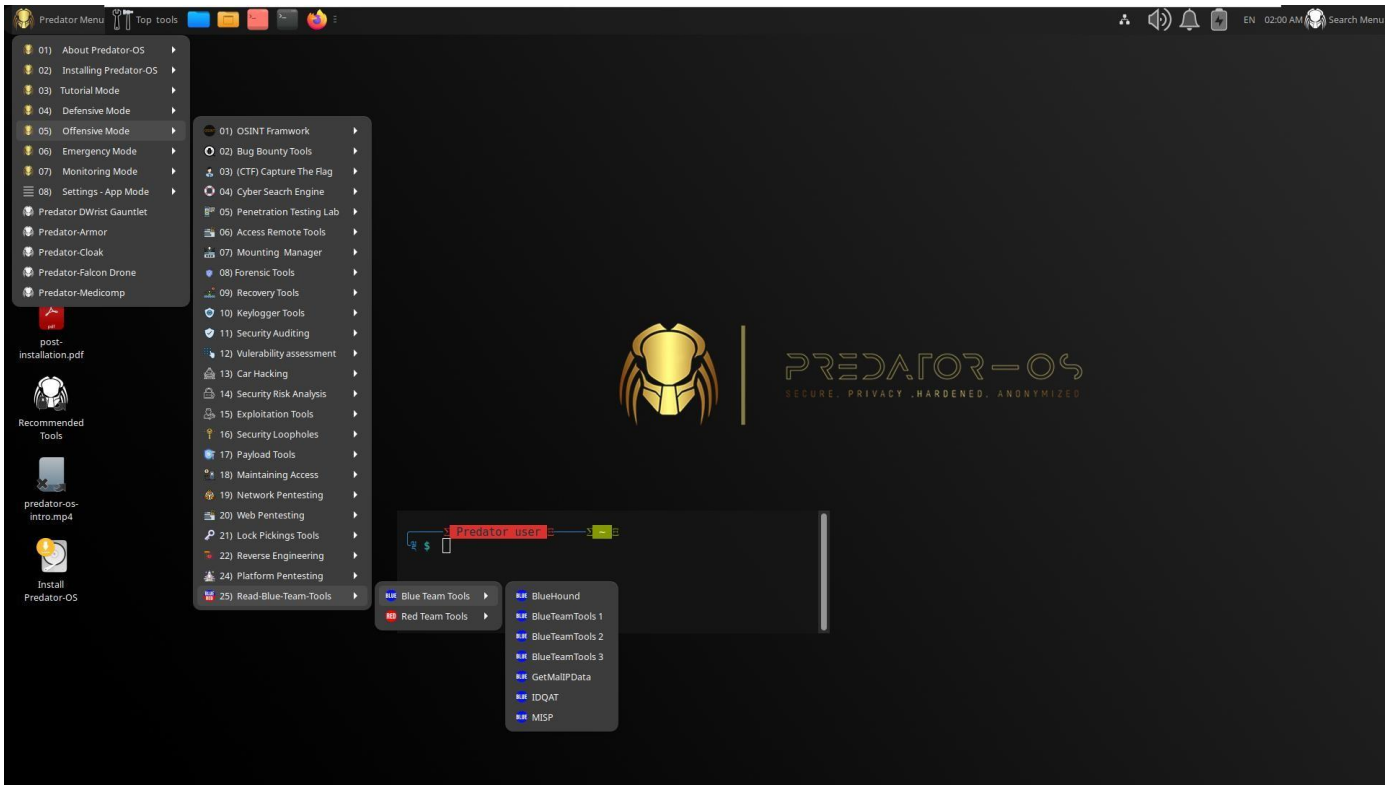
1) Included to more than 2 TB password list.
2) Included to more than 500 lists of the red and blue team tools.
3) Included to more than 200 lists of AWS-cloud tools.
4) Included to more than 10 sets of roadmaps in cybersecurity
5) Included to more than 100 search engines in security and penetration testing 6) Included to more than 300 educational scripts in security and penetration testing
7) Included to more than 100 security training websites and penetration testing for kids.
8) Included to more than 10 tools for running cybersecurity lab and penetration testing 9) Included to more than 40 websites for running a lab in cybersecurity testing. 10) Included to the source of 800 Malware files in 80 different groups (400 MB file) 11) Included to 1000 websites for OSINT.
12) Included to more than 70 online and self-reading websites in cybersecurity.
13) Included to more than 11 offline and self-study training categories in cybersecurity.
14) Included to 600 forensic and reverse engineering tools.
15) Included to more than 6000 Google dorks and exploits as offline

More details:

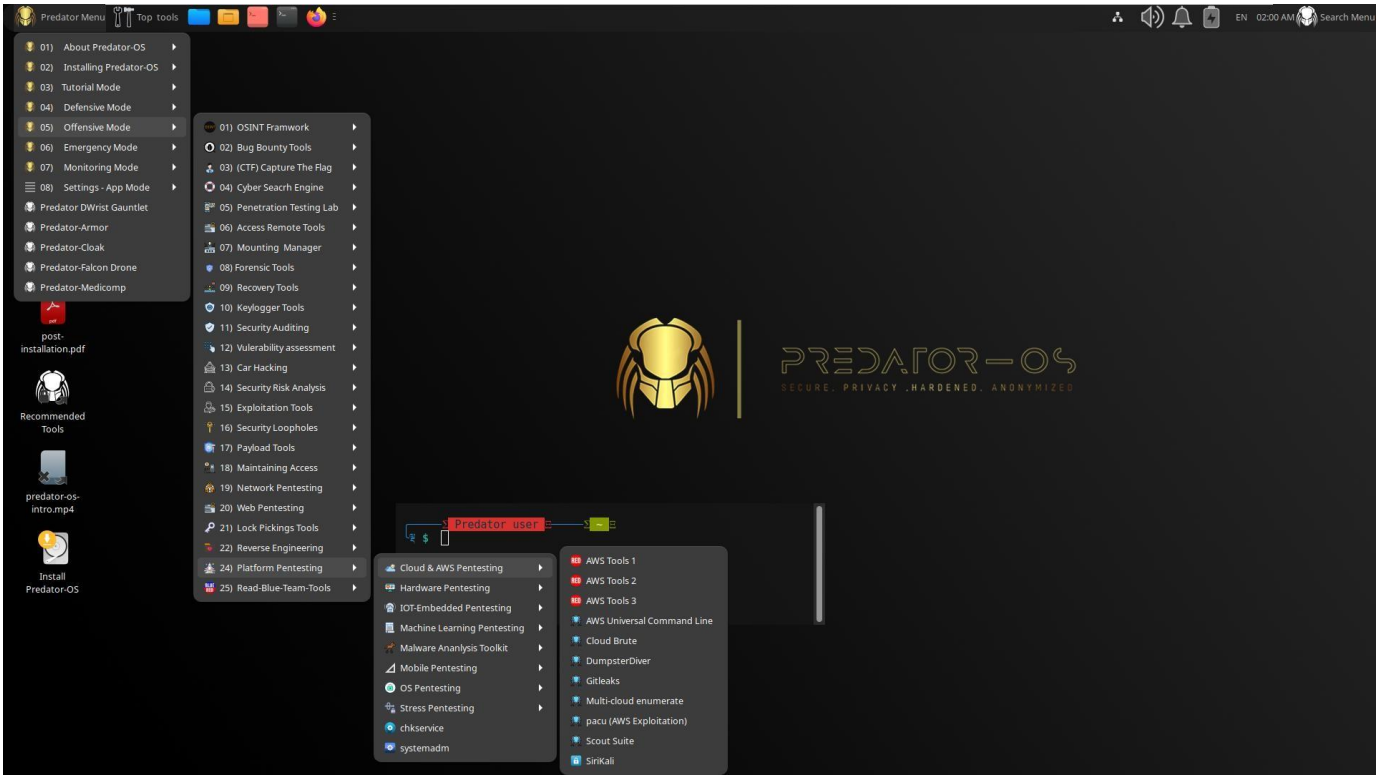
1)Included to more than 2 TB password list.



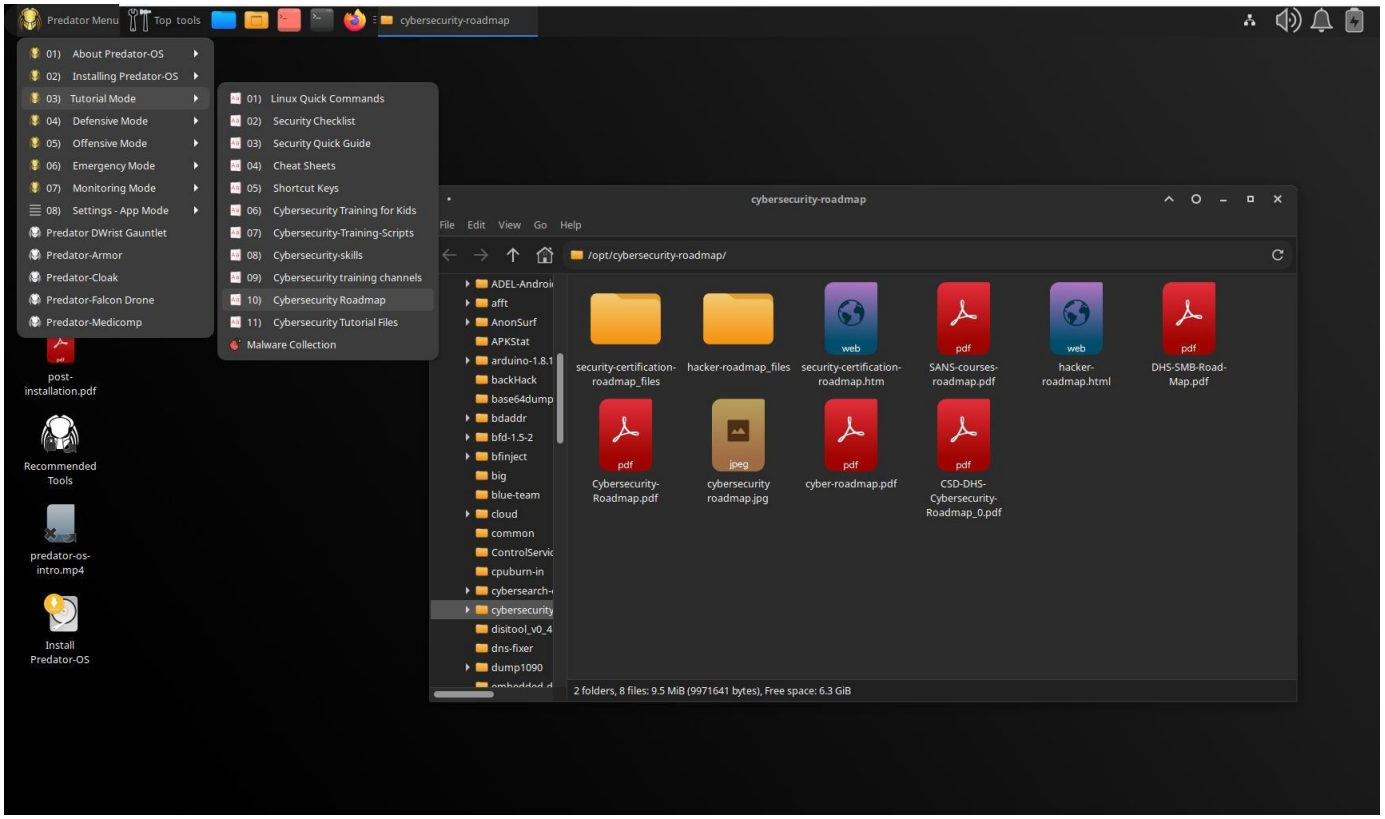
Included to more than 500 lists of the red and blue team tools



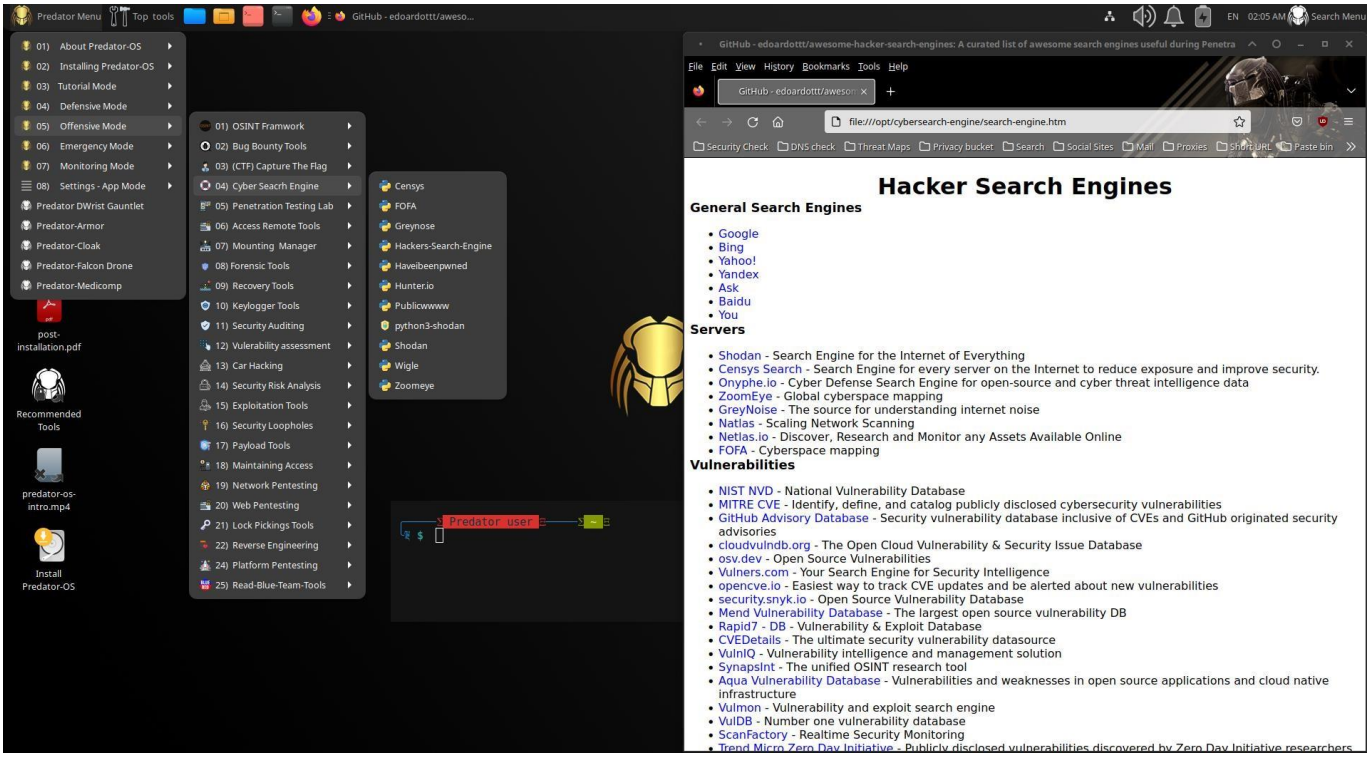
Included to more than 200 lists of AWS-cloud tools



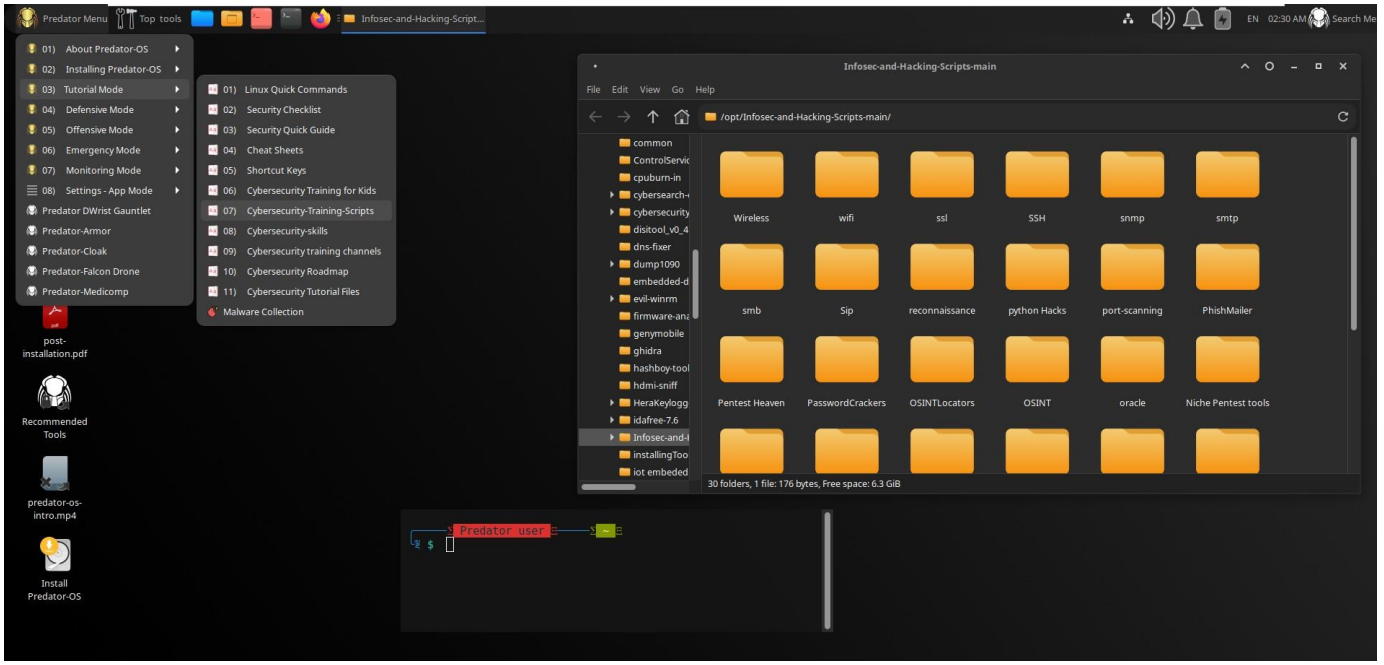
Included to more than 10 sets of roadmaps in cybersecurity



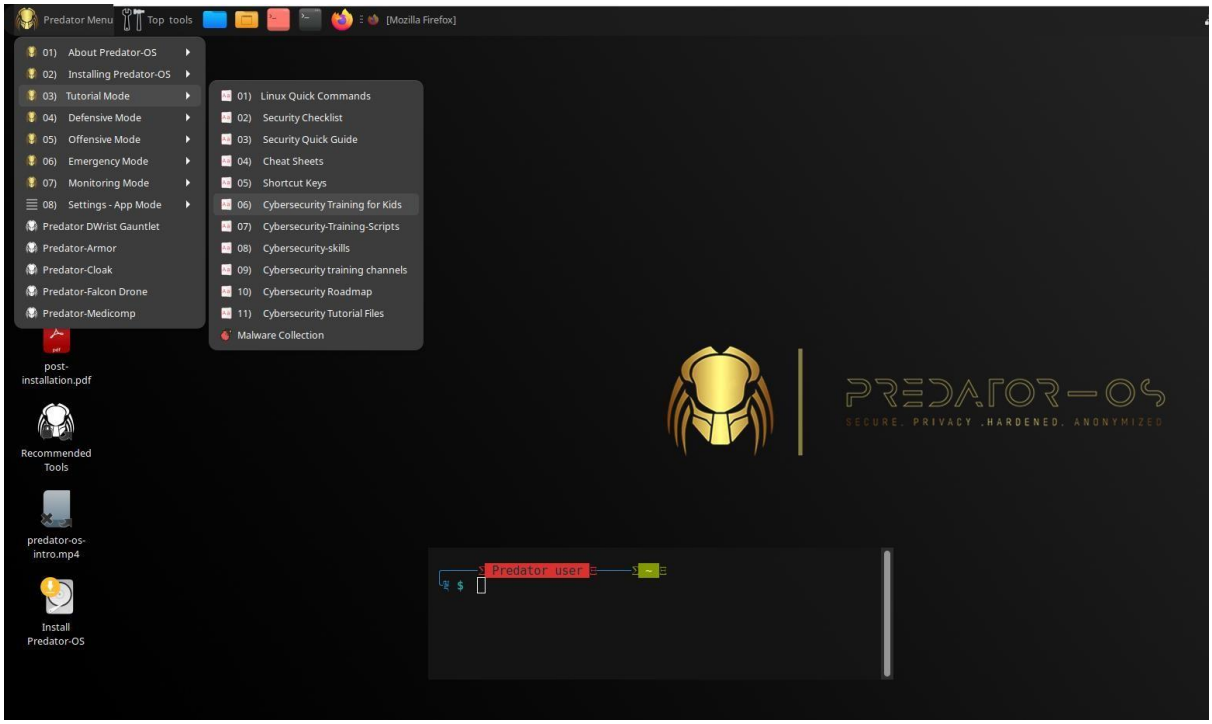
Included to more than 100 search engines in security and penetration testing



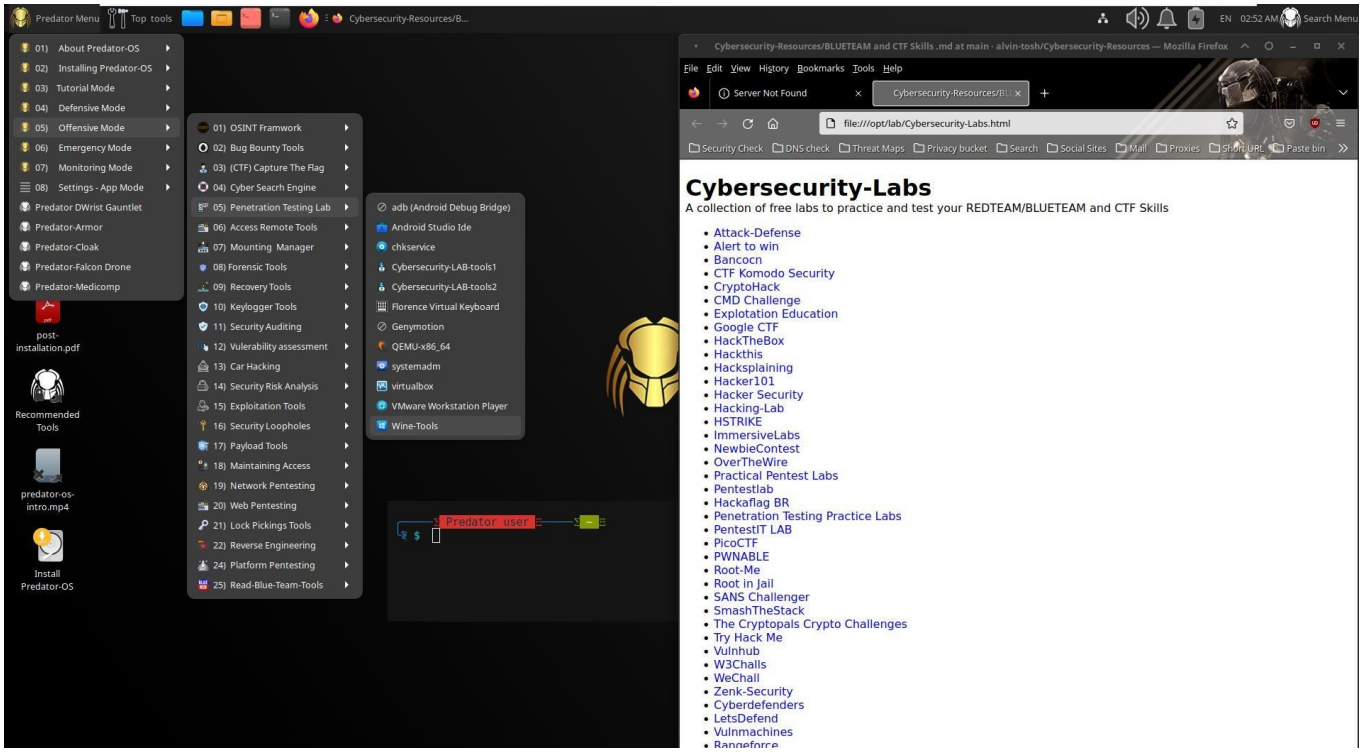
Included to more than 300 educational scripts in security and penetration testing.



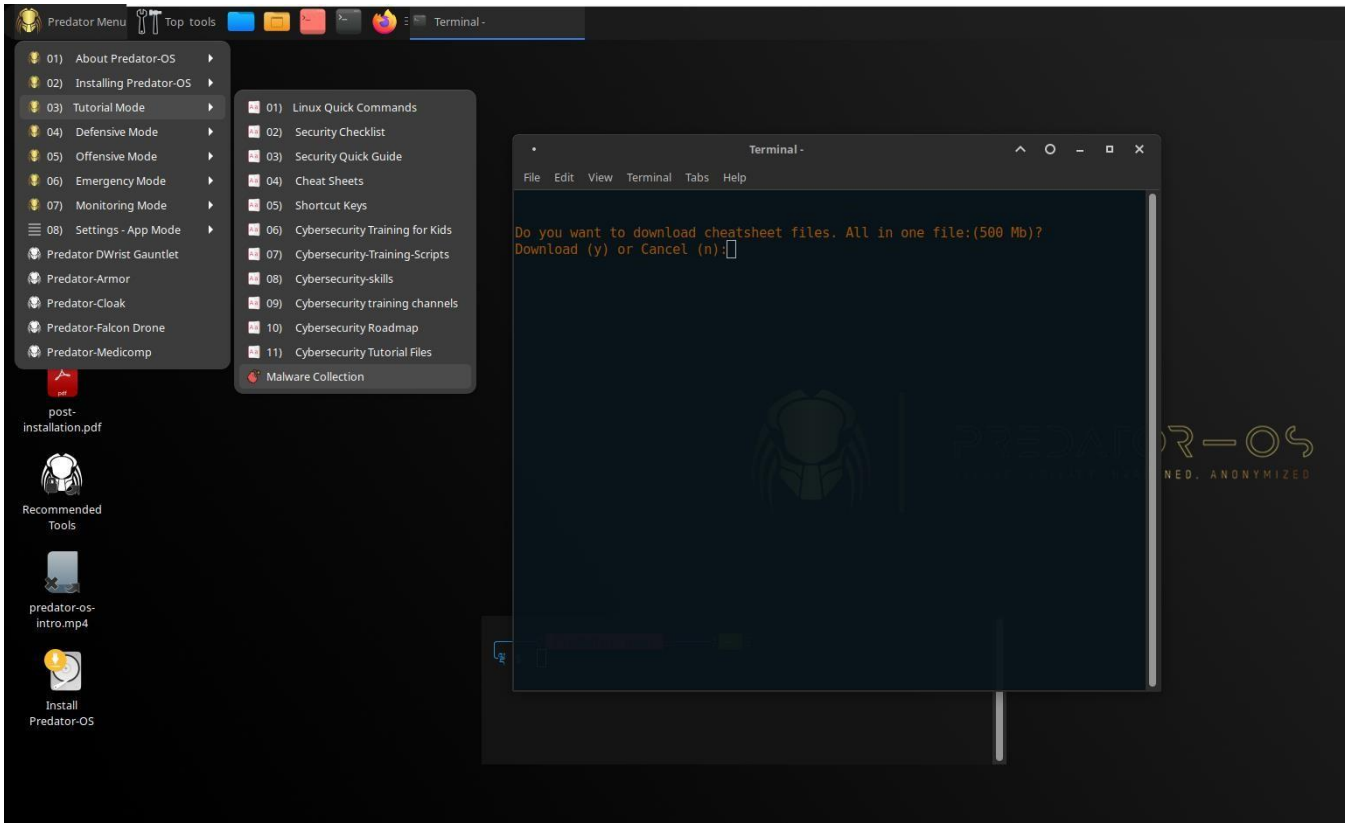
Included to more than 100 security training websites and penetration testing for kids.



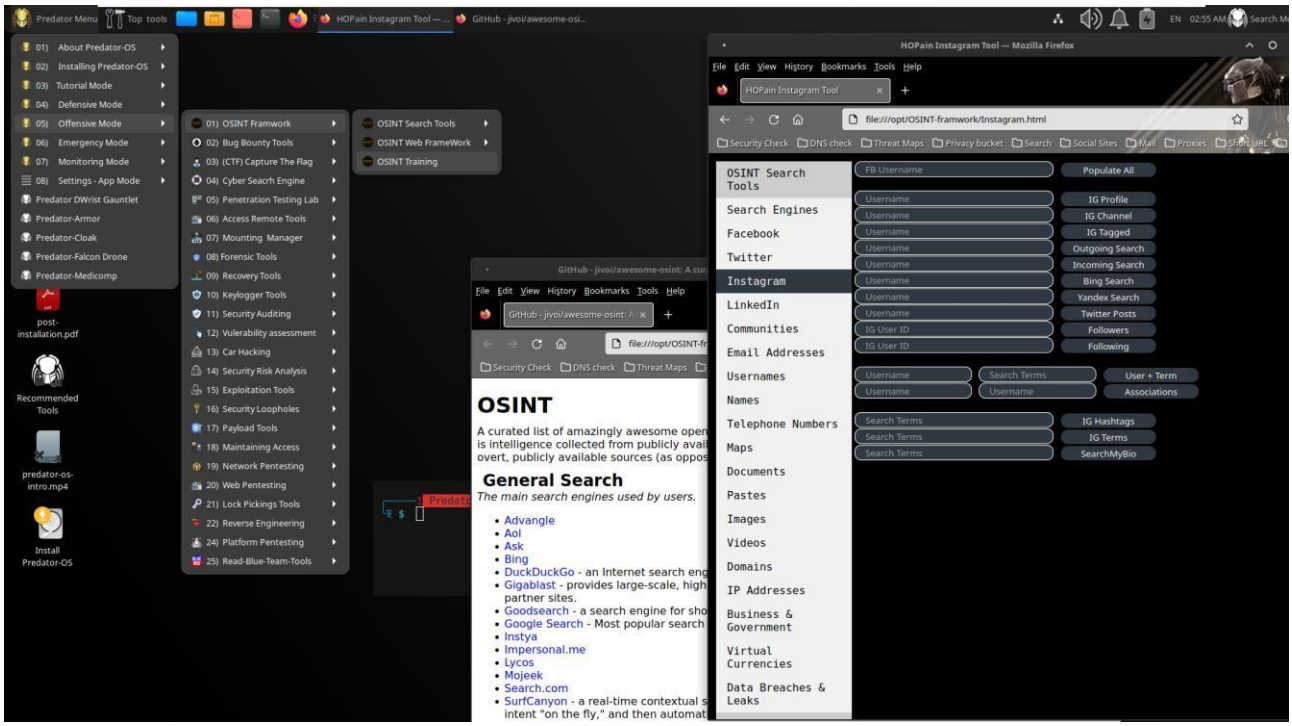
Included to more than 10 tools for running cybersecurity lab and penetration testing



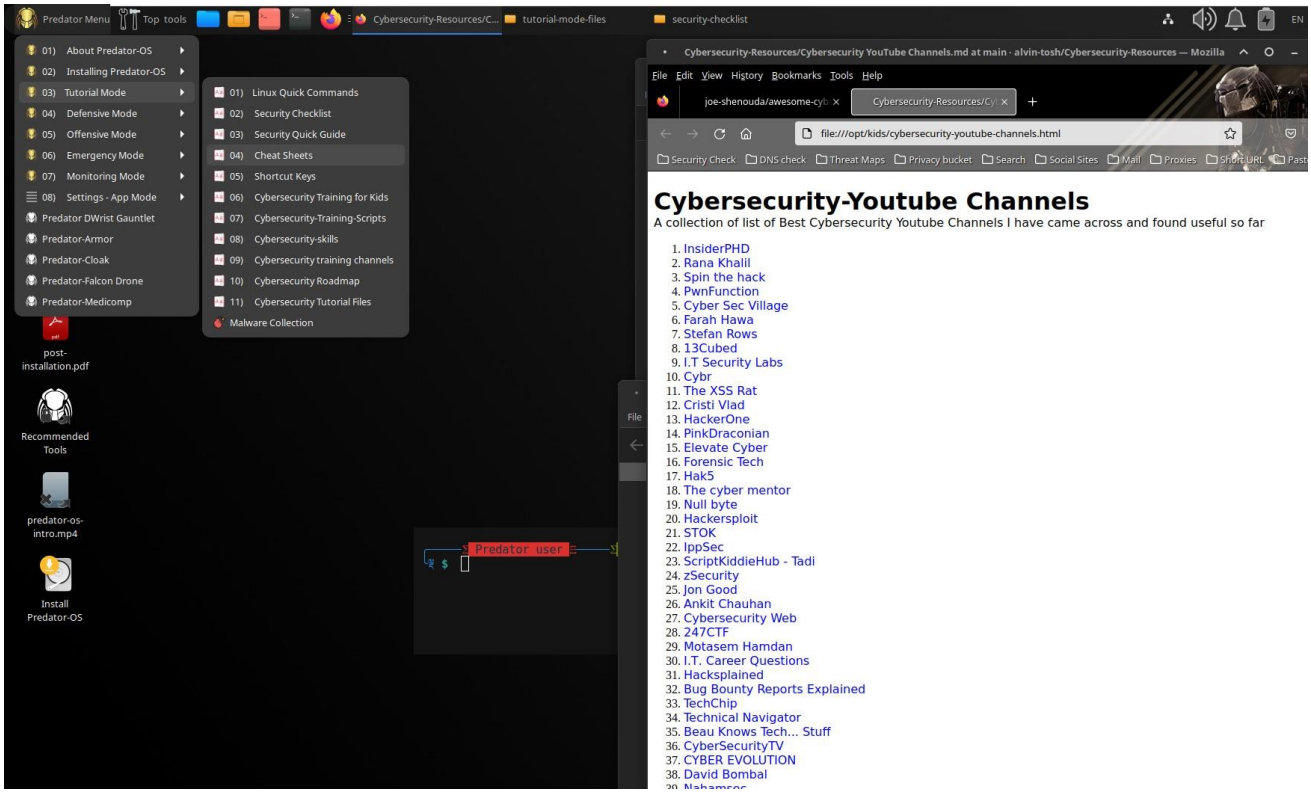
Included to the source of 800 Malware files in 80 different groups (500 MB file)



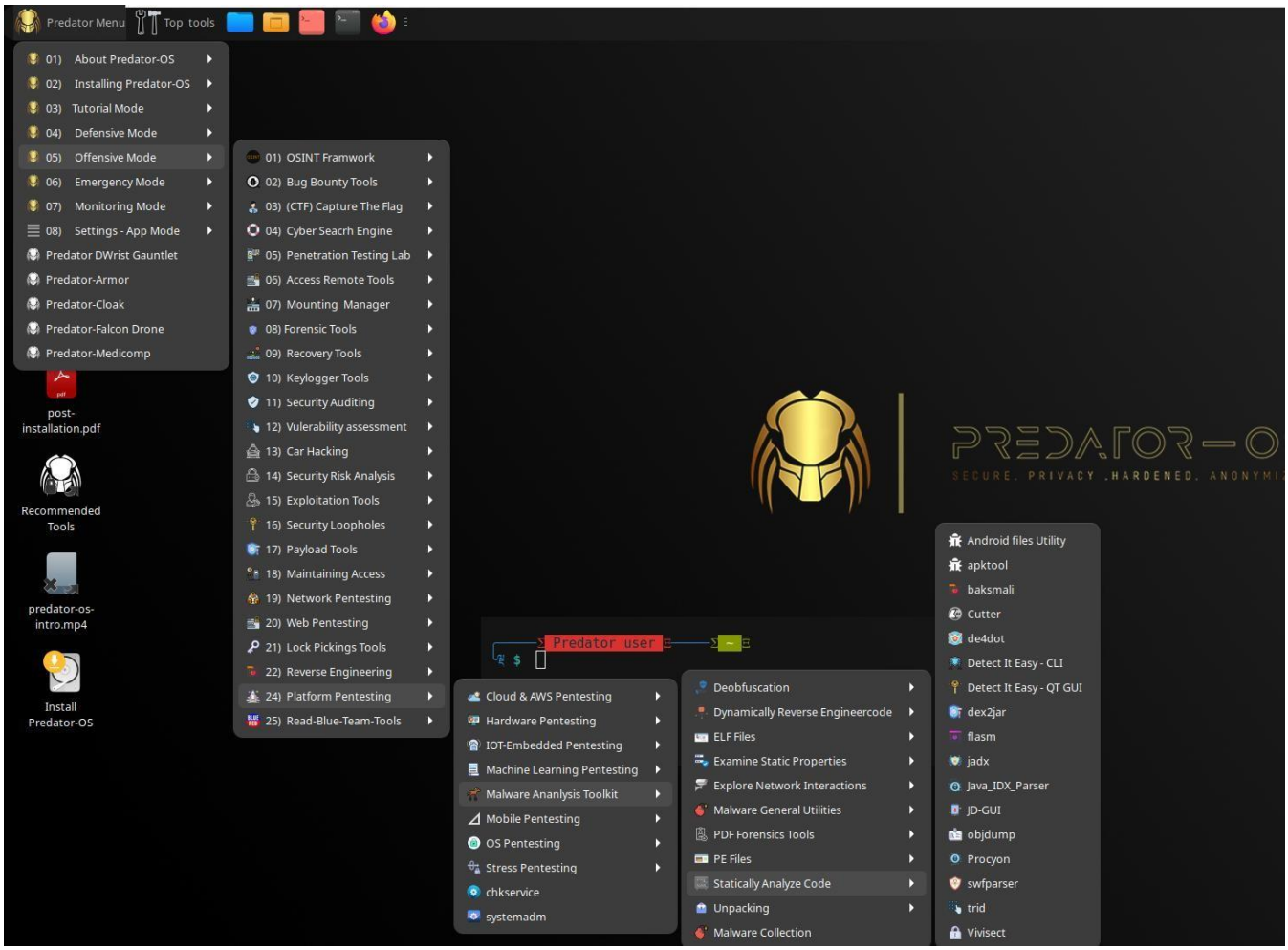
Included to 1000 websites for OSINT.



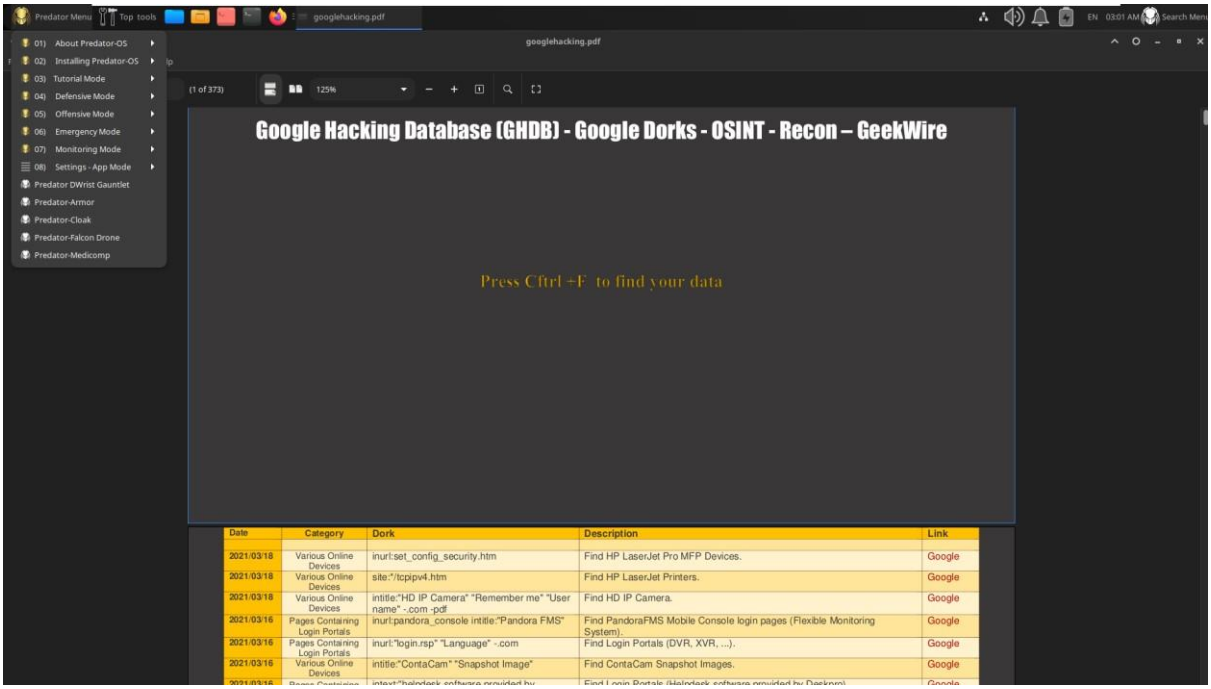
Included to more than 70 online and self-reading websites in cybersecurity.



Included to 600 forensic and reverse engineering tools.



Included to more than 6000 Google dorks and exploits as offline



Improving performance and tuned kernel level and user levels

1) In order to improve the performance of the CPU frequency range has been changed intel_pstate to acpi-cpufreq by default.

2) *The BIOS frequency limitation has been disabled by default in order to improve the performance of the CPU frequency range,*

3)In order to improve the performance of the hard disk and boot time, the watchdog has been disabled by default.

4) Improving the performance of the hard disk by changing the I/O scheduler for SATA, HDD, and NVMe disks.

5) Improved CPU performance by changing the default kernel scheduler to a Linuxzen kernel.

6) Improving network and Internet performance by changing the Bottleneck Bandwidth and Round-trip propagation time (BBR).

7) Improved RAM memory function by changing the randomize_va_space status.

8) Improved virtual memory performance by replacing zswap instead of swap by default.

9) The hardware threads (physical CPU) for each CPU core have been enabled. 10) Improving the paralleling of tasks by allowing independent tasks (running threads) by sharing some processor resources.

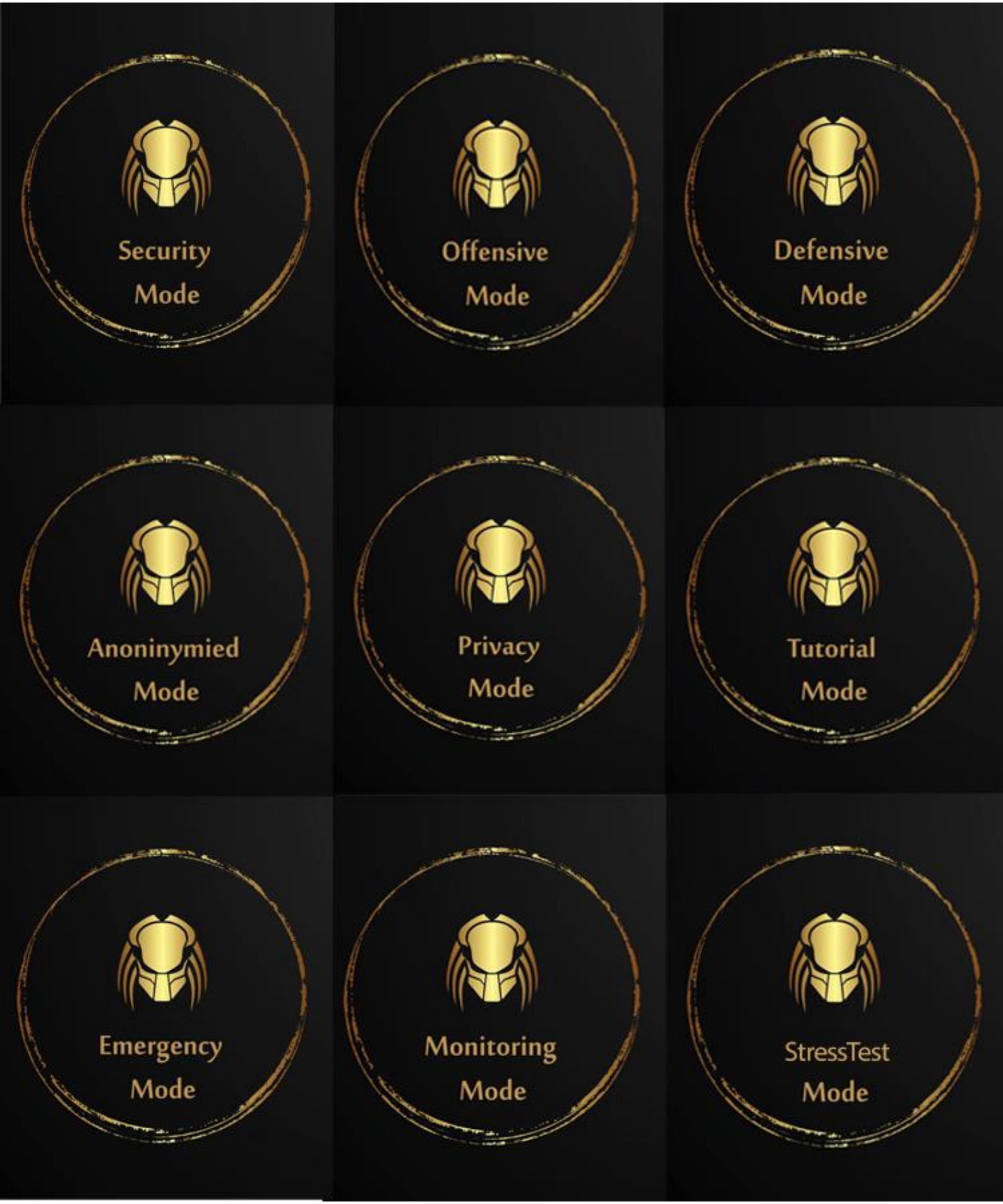
11) Changed power saving mode to performance mode by default, In order to improve the performance of the disk and network IO. 12) All CPU governor frequency has switched in performance.

13) Reducing kernel log-level reports to a low level in order to improve kernel performance and increase security and create silent boot mode.

14) and also, improving TCP performance, increasing inode cache memory, disk cache, improving network and bandwidth parameters, etc.

Operates at 10 different modes:

The Predator-OS has 10 different modes and operates at the following modes for easy and faster access to all tools and it also is possible to change Linux Predator at: defensive, offensive, privacy, hardened, secured, settings and pretesting modes quickly.


Security Mode
Offensive Mode
Defensive Mode
Anoninymied Mode
Privacy Mode
Tutorial Mode
Emergency Mode
Monitoring Mode
StressTest Mode

History

Predator-OS Linux started in 2021, initially it was supposed to be a privacy and anonymous Linux. It was due to the many problems users reported in the use of other Linux distributions in the field of security and the stopping of many distributions, this distribution was created.

At first, it was decided to make Predator-OS based on Debian. But due to having pre-installed security tools; as well as Debian being a rolling release, the distribution would have many updates and upgrades. And for this issue, I created the distribution in the fix release, so that updates can be done periodically. The best option for Debian stable distribution was the LTS version, which is known for its quality, stability, and wide selection of available software.

The first version of Predator-OS (version 1.0) was released nine months later after some issues and was based on the Debian stable Mini 18.04 at the time. In that first year of development, the focus was on security, privacy, anonymous distribution along with penetration testing tools.

After version 1.0, Predator-OS releases an update annually with new features and improved hardware support. I considered the PLASMA desktop for it because of its simplicity and speed, as well as better customization than other desktops. In version, 2 of the Predator-OS distribution, about 25 new features were released that switched from Debian stable 18.04 to 20.04.

In version 2, many things were considered, both in distribution optimization, turning, and tools.

I was trying to create a distribution with the best performance in several working modes in the field of security, which can cover all cases according to the extent of security. Finally, I created mods for the Predator-OS and it now works in 9 modes.

In version 2.5 of this distribution, it was released with 50 modifications and new features. Most of the focus of version 2.5 was on performance.

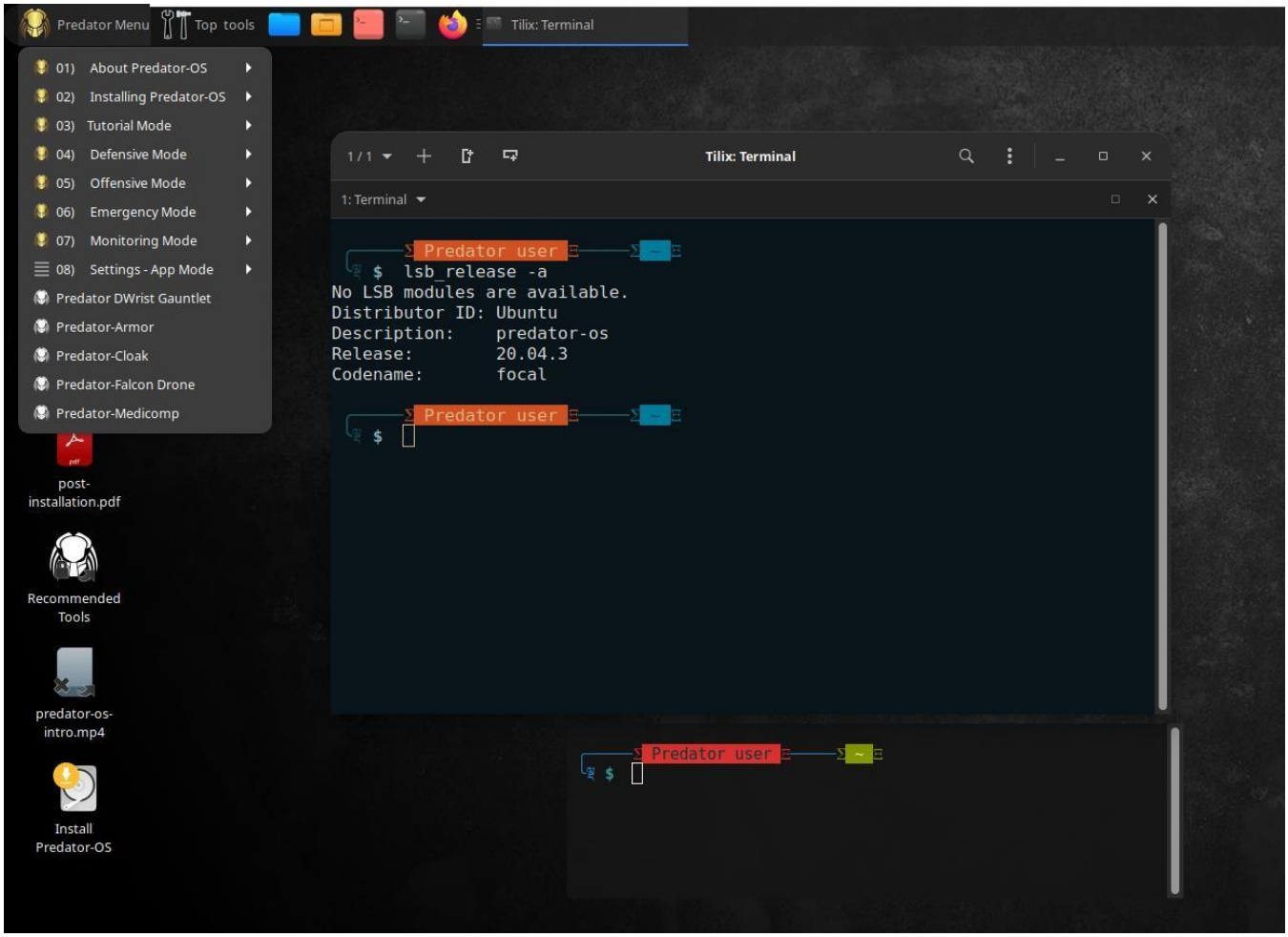
In version 3, the distribution will be towards artificial intelligence and the use of intelligent defense mechanisms in attack and defense mode.

Based on Debian stable

Debian stable is open-source software that was developed by Canonical in October 2004. Debian stable is a Linux-based operating system. It is designed for computers, smartphones, and network servers. A UK based company called Canonical Ltd. develops the system. All the principles used to develop the Debian stable.

LTS Release

It is a fixed-release distribution based on Debian stable mini 22.04.3 with the PLASMA desktop and kernel 5.18 LTS. A Polymorphic Security Platform that be able to run as live medium and you can install it on hard disk.



Why LTS version?

LTS option allows users to stick with a particular version for an extended period of the software without worrying about the quality degrading due to technical issues or security vulnerabilities. LTS releases are more secure, stable, and hence reliable. In addition, all LTS releases get software updates and standard support for up to five years from the Canonical team. For servers, software development

PCs, and other critical systems, it is highly recommended that you use the LTS version.

https://Predator-OS.ir

Rolling Release

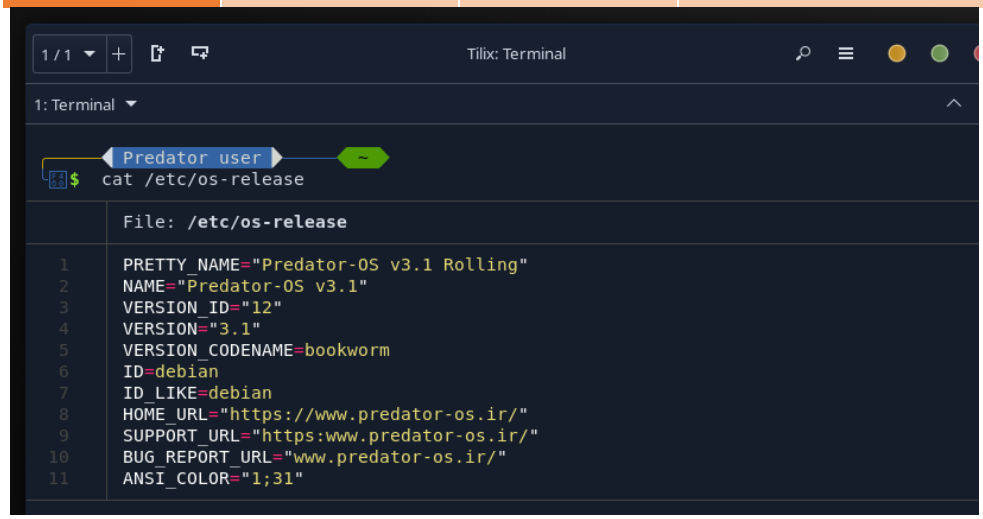
Another edition of rolling release is available that is use daily build source list.

https://Predator-OS.ir

Predator-OS Lifecycle

Predator-OS Linux is released with new update and features every year on the first day of the year.

Predator-OS Version	Code name base Debian stable	Release date	Based on
3.1	Debian bookworm	2024-04-04	Debian Stable bookworm
3.0	jammy	2023-11-01	ubuntu stable 22.04 LTS (jammy)
2.5	Focal Fossa	2023-01-01	ubuntu stable 20.04 LTS (Focal Fossa)
2.0	Cosmic Cuttlefish	2022-01-01	ubuntu stable 18.10 (Cosmic Cuttlefish)
1.0	Bionic Beaver	2021-01-01	ubuntu stable 18.04 LTS (Bionic Beaver)

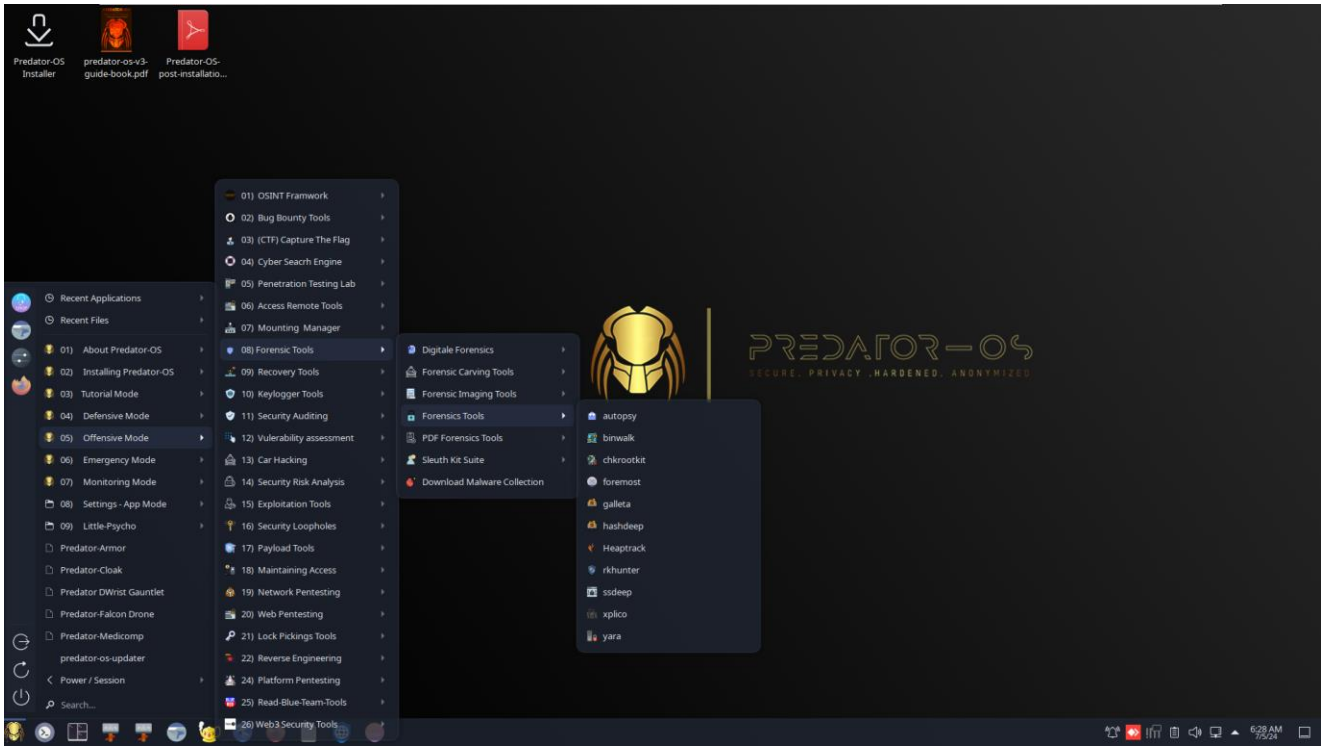


Multi-Platform Operating System

Currently, it officially supports amd64 hardware release architectures. I will release ARM architecture soon. Furthermore, with more than 1300 packages, the pre-installed software can meet almost any need, whether at home or in the enterprise.

Number of tools

When you boot up Predator-OS, you will quickly notice that a security learning process into a variety of different contexts and activities organizes the main menu. Predator Linux has neat 1000 pre-installed tools for cybersecurity and pentesting, and also system administrators. These tools are split into 40 categories.



Kernel custom

The predator Linux kernel has been recompiled and tuned as much as possible to optimize the system, secure, hardened and anonymous it and have the latest patches and hardware support and firmware.

Improving performance and tuned kernel level and user levels:

1) In order to improve the performance of the CPU frequency range has been changed intel_pstate to acpi-cpufreq by default.
2) The BIOS frequency limitation has been disabled by default in order to improve the performance of the CPU frequency range,
3) In order to improve the performance of the hard disk and boot time, the watchdog has been disabled by default.
4) Improving the performance of the hard disk by changing the I/O scheduler for SATA, HDD, and NVMe disks.
5) Improved CPU performance by changing the default kernel scheduler to a Linux-zen kernel. 6) Improving network and Internet performance by changing the Bottleneck Bandwidth and Roundtrip propagation time (BBR).
7) Improved RAM memory function by changing the randomize_va_space status.
8) Improved virtual memory performance by replacing zswap instead of swap by default.
9) The hardware threads (physical CPU) for each CPU core have been enabled.
10) Improving the paralleling of tasks by allowing independent tasks (running threads) by sharing some processor resources.
11) Changed power saving mode to performance mode by default, in order to improve the performance of the disk and network IO.
12) All CPU governor frequency has switched in performance.
13) Reducing kernel log-level reports to a low level in order to improve kernel performance and increase security and create silent boot mode.
14) and also, improving TCP performance, increasing inode cache memory, disk cache, improving network and bandwidth parameters, etc.

The Predator-OS Developers

The Predator-OS has been designed and developed by Hossein seilani who is the developer of Emperor-OS Linux too. And the website and documents are designed by him as well.

He is originally from Iran.

Predator-OS News:

Most important news about Predator-OS:

https://Predator-OS.ir

Hardware requirements

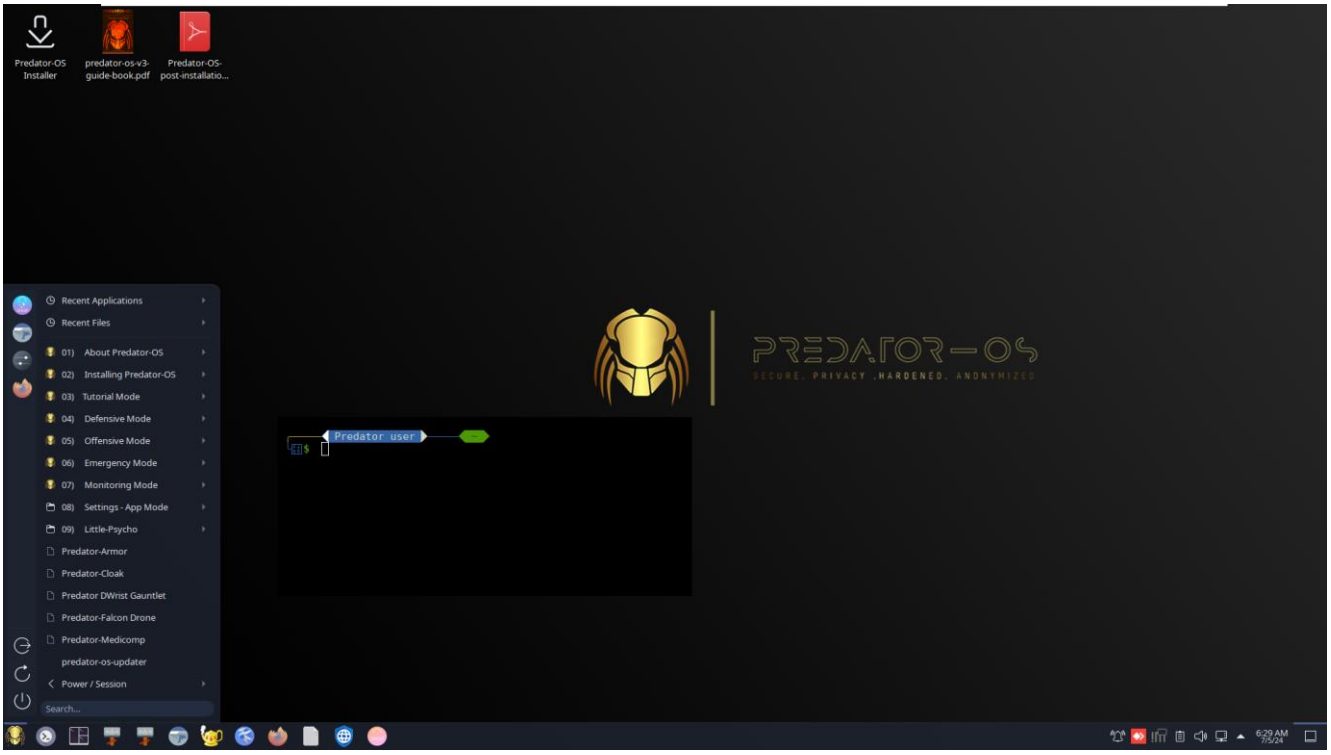
Predator-OS Linux is only compatible with 64-bit processors.

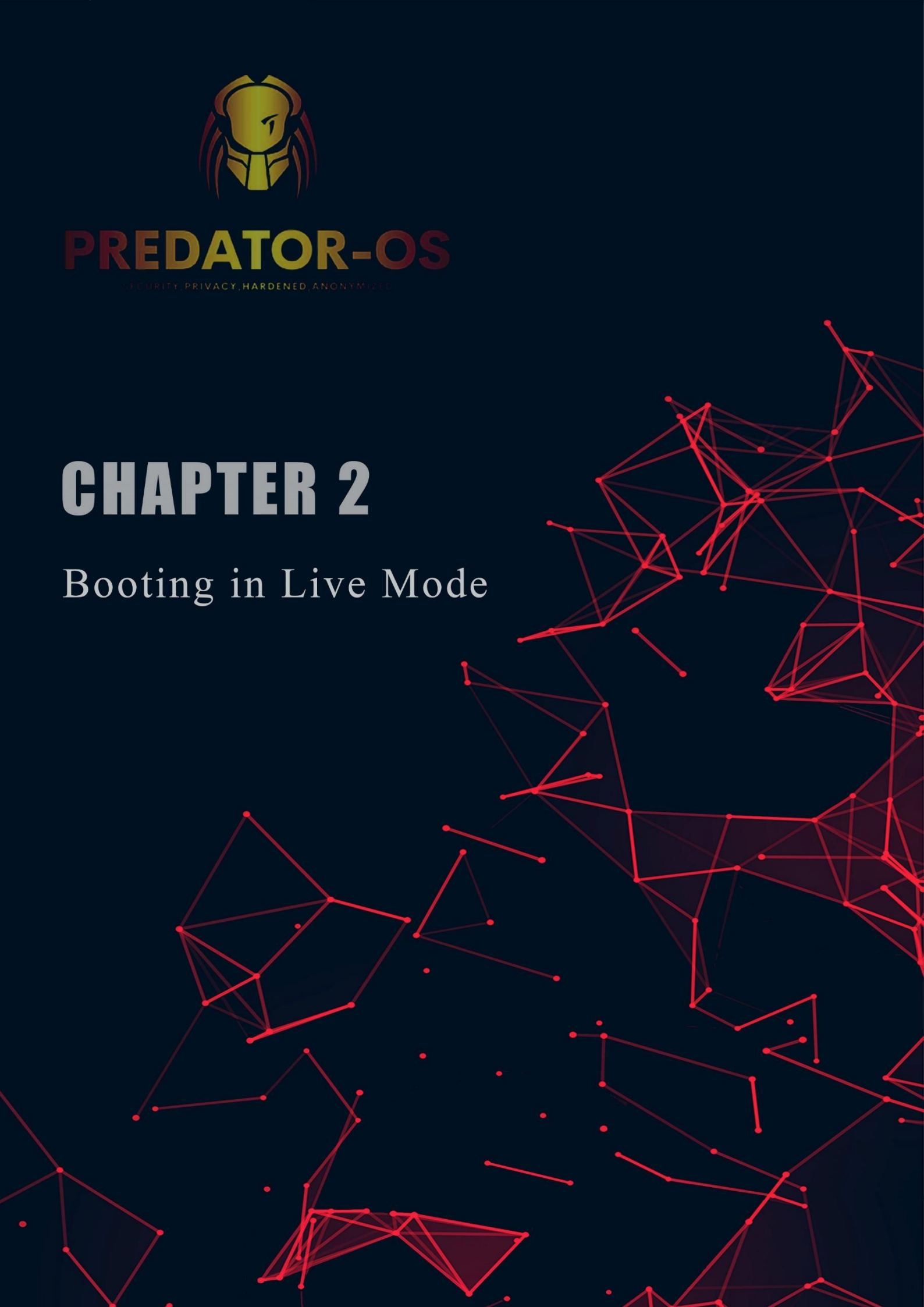
Recommended system requirements:

- 2 GHz dual-core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Internet access is helpful
- Either a DVD drive or a USB port for the installer media

Full compatibility on live system

You can run predator-so on live mode and you will have all features of it in live mode.





CHAPTER 2

Booting in Live Mode

Chapter 2

Booting in Live Mode

Booting a Predator-OS ISO Image in Live Mode

Running Predator-OS directly from either a USB stick or a DVD is a quick and easy way to experience how Debian stable works for you and how it works with your hardware. Most importantly, it does not alter your computer's configuration in any way and a simple restart without the USB stick or DVD is all that's needed to restore your machine to its previous state. A live CD (also live DVD, live disc, or live operating system) is a complete bootable computer installation including operating system, which runs directly from a CD-ROM or similar storage device into a computer's memory, rather than loading from a hard disk drive. A live CD allows users to run an operating system for any purpose without installing it or making any changes to the computer's configuration. Live CDs can run on a computer without secondary storage, such as a hard disk drive, or with a corrupted hard disk drive or file system, allowing data recovery.

With a live Predator-OS, you can do almost anything you can from an installed Debian stable:

Safely browse the internet without storing any history or cookie data.

Access files and edit files stored on your computer or USB stick.

Create new office suite documents and save them remotely.

Fix broken configurations to get a computer running again.

The concept of live booting is actually quite simple. With a live Linux distribution (not all distributions come in "live" flavors), you can boot your machine from either a CD/DVD disk or from a USB flash drive and choose to try out the operating system without making any changes to your hard drive.

How this works is by running the entire system from volatile memory (RAM). The operating system and all programs are usable, but run from memory. Because of this, you can boot the live system, test/use it for as long as you need, and then reboot the system (remembering to remove the live media) to return to your original system. Live distributions can be used for several purposes: Testing a Linux distribution: This is the best way to see if Linux is for you. Testing hardware: If you are unsure if your hardware will work with a Linux distribution, run it live and find out.

Live distributions also form a collection of very important tools that handle crucial tasks, such as:

- Data recovery
- System recovery
- Rescue and repair
- PC Forensics
- Boot repair

Predator-OS Linux has the functionality of being installed on a hard drive. The installation is possible both in UEFI and in MBR. Predator-OS Linux is 100% ompatible with classic BIOS called LEGACY and a UEFI BIOS.

BIOS vs. UEFI Traditional PC firmware was called the BIOS, for Basic Input/Output System. Over the last decade, however, BIOS has been supplanted by a more formalized and modern standard, the Unified Extensible Firmware Interface (UEFI). You will often see UEFI referred to as "UEFI BIOS," but for clarity, we will reserve the term BIOS for the legacy standard in this book. Most systems that implement UEFI can fall back to a legacy BIOS implementation if the operating system they are booting does not support UEFI. UEFI is the current revision of an earlier standard, EFI. References to the name EFI persist in some older documentation and even in some standard terms, such as "EFI system partition." In all but the most technically explicit situations, you can treat these terms as equivalent.

Default username and password

If you need the user or password on Predator-OS, use the following defaults:

Login: user
Password: user

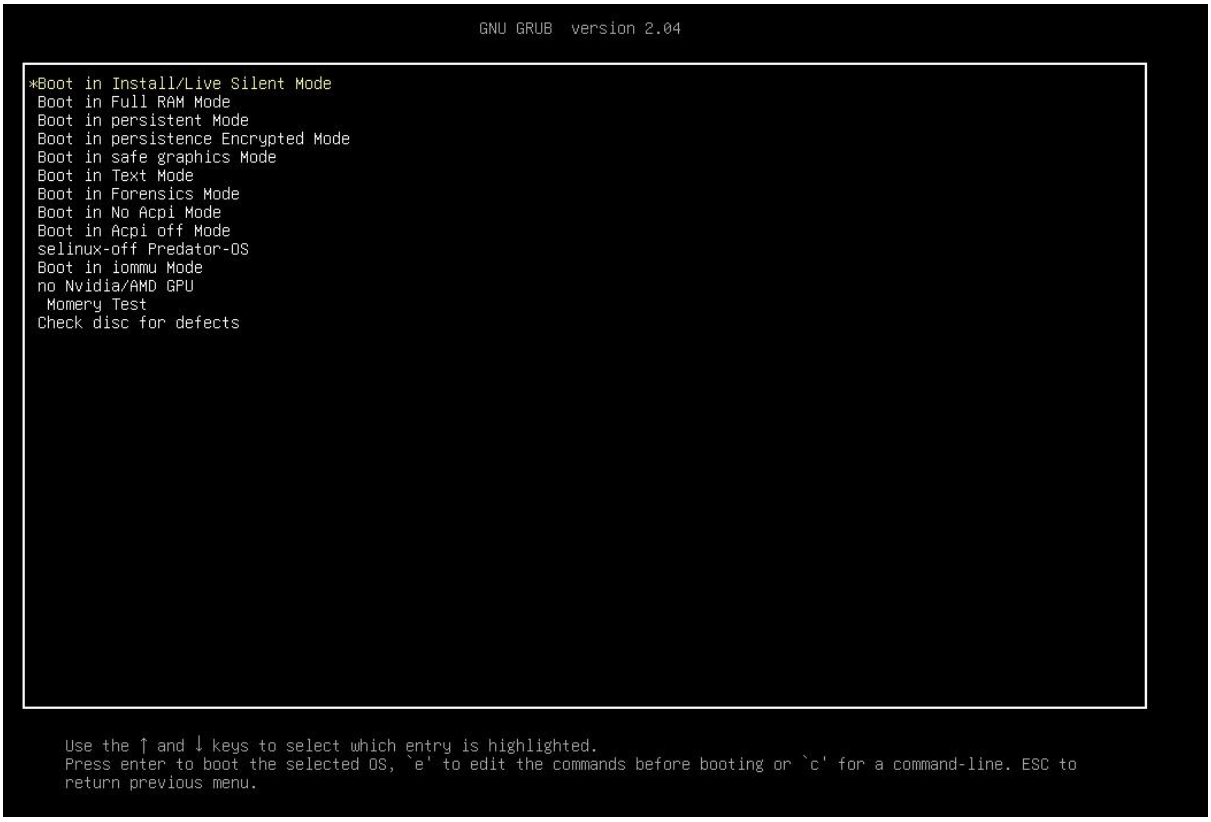
Live options

Predator-OS Linux Grub boot menu has several options:

Boot in Live/install Mode
Boot in Direct Install
Boot in Full RAM Mode
Boot in persistent Mode
Boot in persistence Encrypted Mode
Boot in safe graphics Mode
Boot in Text Mode
Boot in Forensics Mode
Boot in No Acpi Mode
Boot in iommu Mode
Boot in Acpi off Mode
Boot with no Nvidia/AMD GPU
Boot in Memory Test boot the
first hard disk
Boot in no Nvidia/AMD GPU

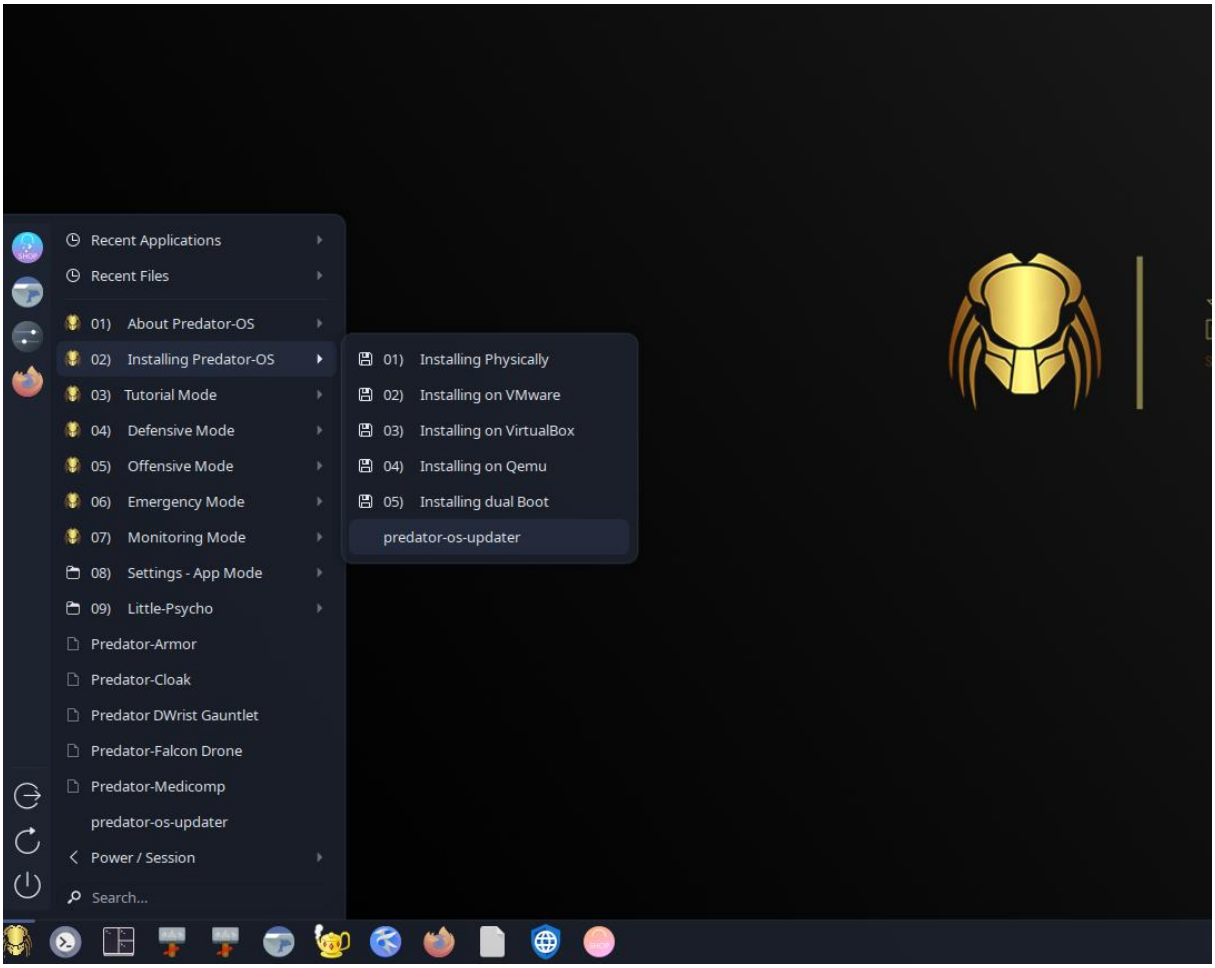
Grub boot menu in MBR boot(legacy)

If you boot the usb boot on mbr mode, you will see this image.



Grub boot menu in UEFI boot(GPT)

If you boot the usb boot on uefi mode, you will see this image.
Access to installation guide in this menu in the live mode:


Recent Applications
Recent Files
01) About Predator-OS
02) Installing Predator-OS
03) Tutorial Mode
04) Defensive Mode
05) Offensive Mode
06) Emergency Mode
07) Monitoring Mode
08) Settings - App Mode
09) Little-Psycho
Predator-Armor
Predator-Cloak
Predator DWrist Gauntlet
Predator-Falcon Drone
Predator-Medicomp
predator-os-updater
Power / Session
Search...
01) Installing Physically
02) Installing on VMware
03) Installing on VirtualBox
04) Installing on Qemu
05) Installing dual Boot
predator-os-updater

PREDATOR-OS

SECURITY, PRIVACY, HARDENED, ANONYMIZED

CHAPTER 3

Pre-Installing

Chapter 3

Pre-installing Predator-OS

Create USB boot device:

By using a bootable predator USB stick, you can:

1) Install or upgrade Predator-OS:
2) Try it without installing predator desktop and without touching your PC configuration.
3) Use all predator's tools installed by default on the USB stick to repair or fix a broken configuration.

Create a bootable USB stick with Rufus on Windows

This tutorial will show you how to create a bootable USB stick on Microsoft Windows using Rufus.

Requirements

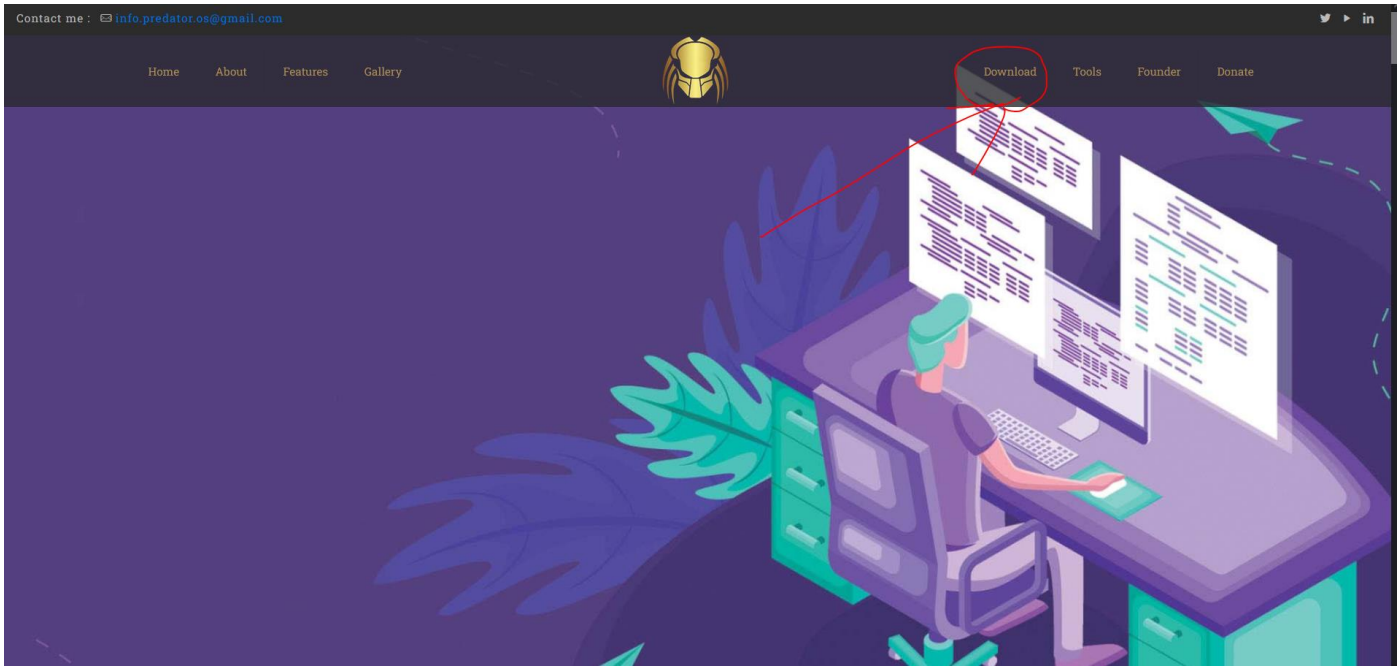
You will need:

- A 4GB or larger USB stick/flash drive
- Microsoft Windows
- Rufus, a free and open source USB stick writing tool
- A Debian stable ISO file. See Get Debian stable for download links

1. Download Predator-OS ISO file.

Go to the main page:

https://Predator-OS.ir/

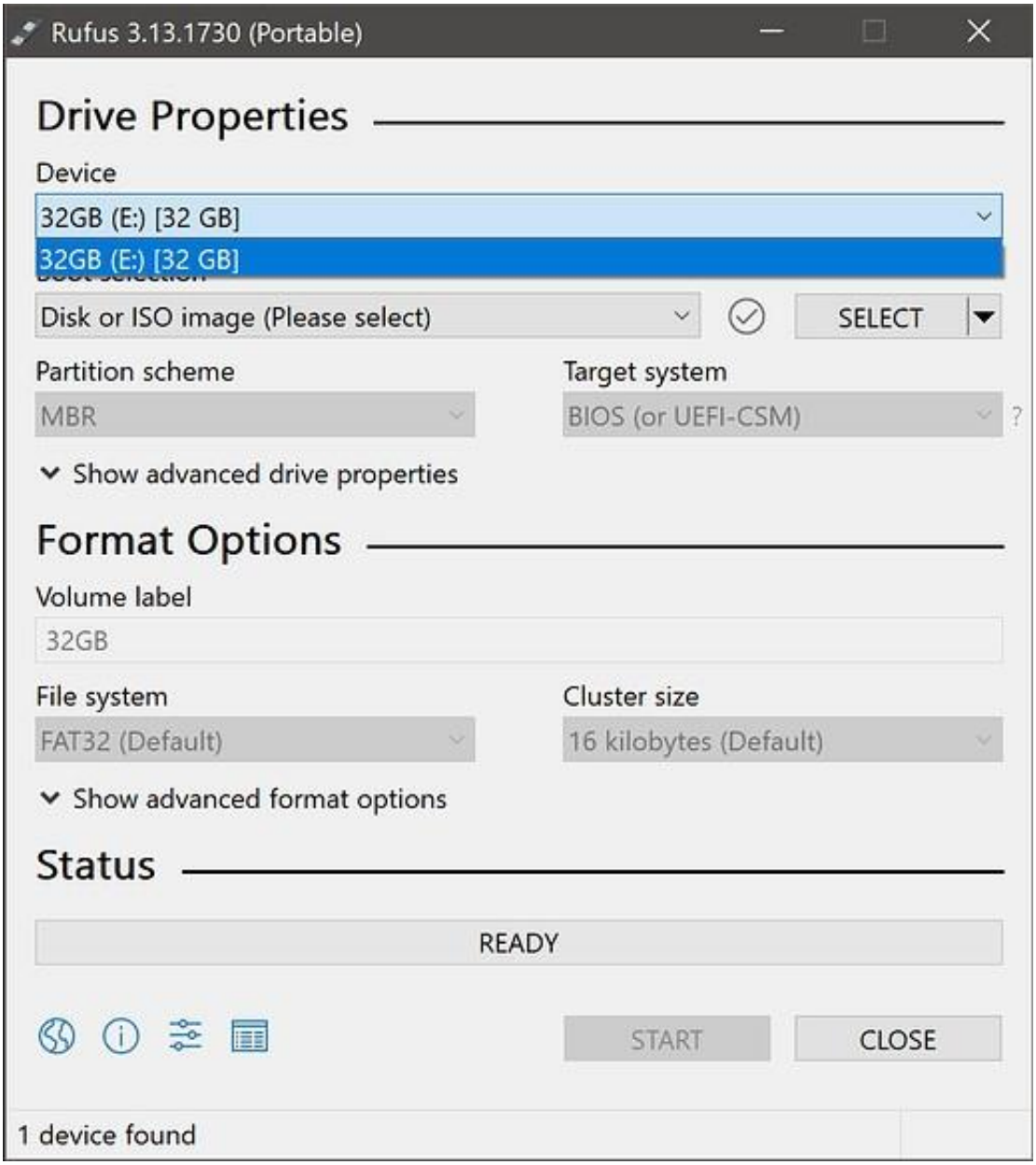


Take note of where your browser saves downloads: this is normally a directory called 'Downloads' on your Windows PC. If using Windows XP or Vista, download Rufus.

USB selection

Perform the following to configure your USB device in Rufus:

1. Launch Rufus
2. Insert your USB stick
3. Rufus will update to set the device within the Device field
4. If the Device selected is incorrect (perhaps you have multiple USB storage devices), select the correct one from the device field's drop-down menu



You can avoid the hassle of selecting from a list of USB devices by ensuring no other devices are connected.

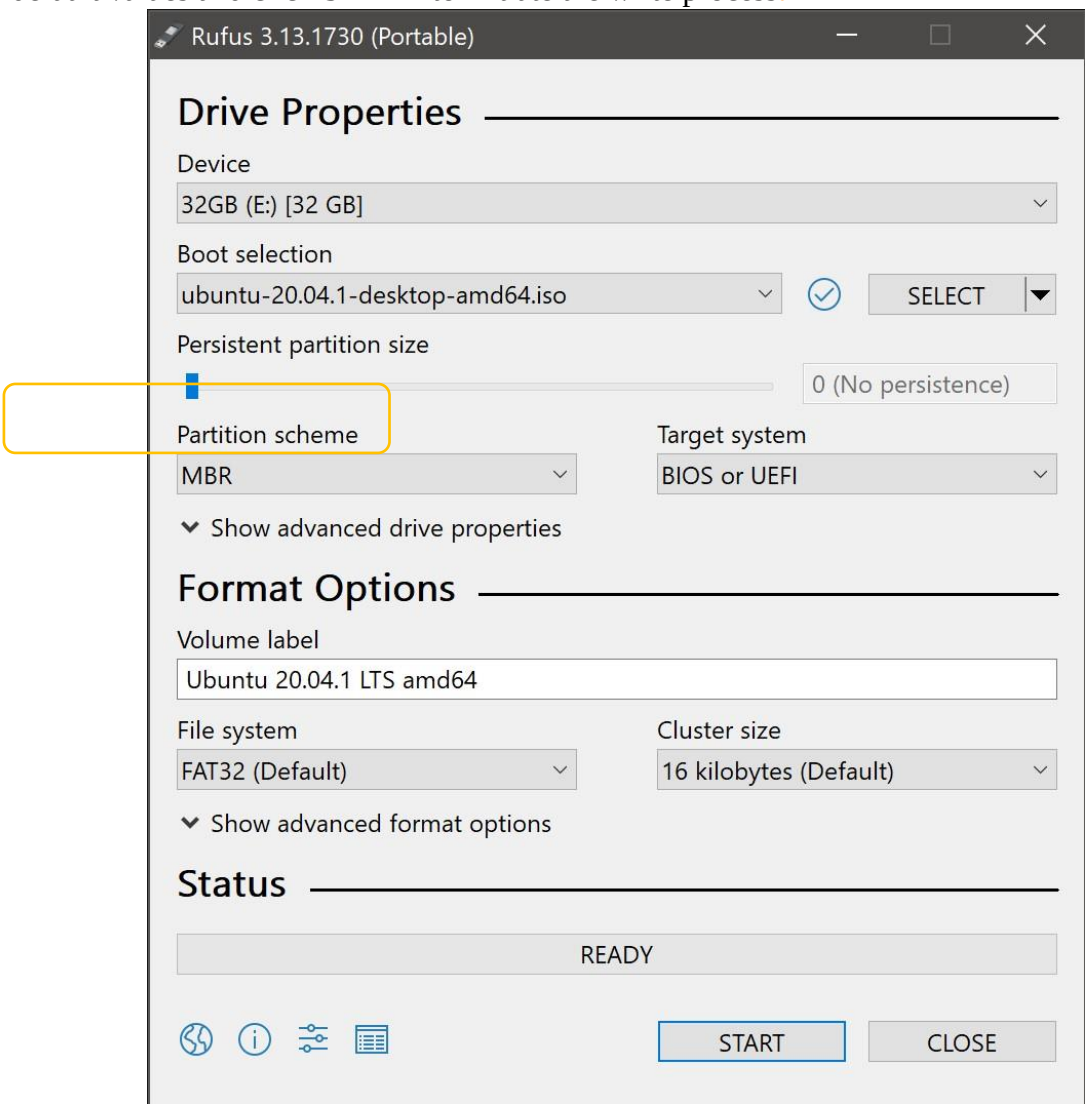
Select the predator-os stable ISO file

To select the Debian stable ISO file, you downloaded previously, click the SELECT to the right of "Boot selection". If this is the only ISO file present in the Downloads folder, you will only see one file listed.

Select the appropriate ISO file and click on Open.

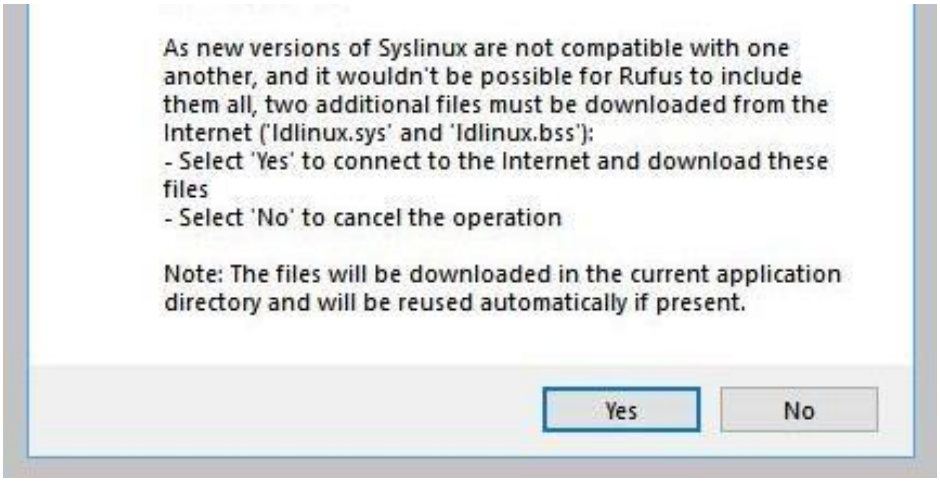
Write the ISO

The Volume label will be updated to reflect the ISO selected. Leave all other parameters with their default values and click START to initiate the write process.



Additional downloads

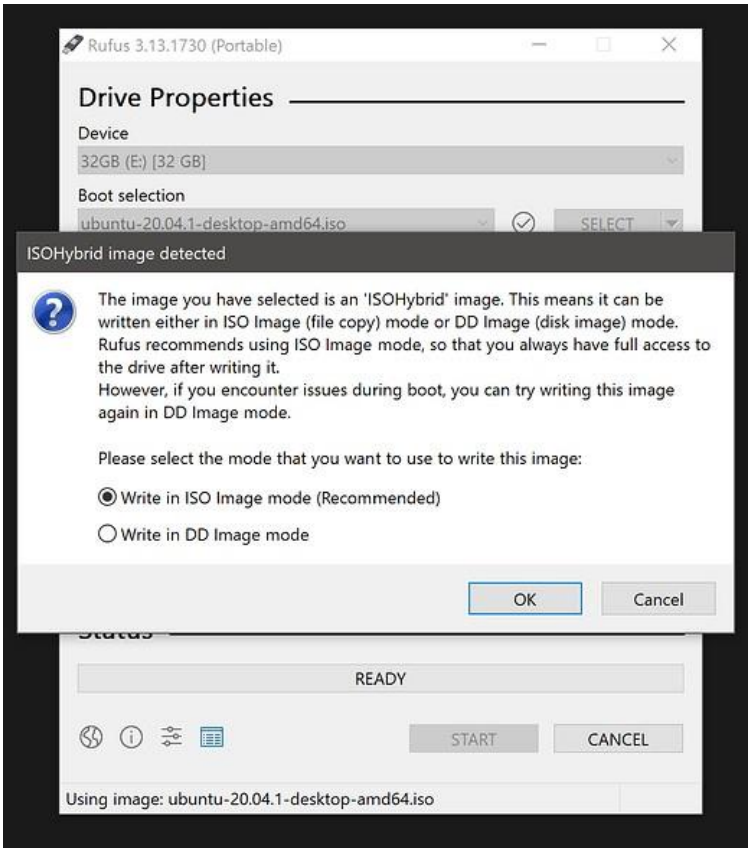
You may be alerted that Rufus requires additional files to complete writing the ISO. If this dialog box appears, select Yes to continue.



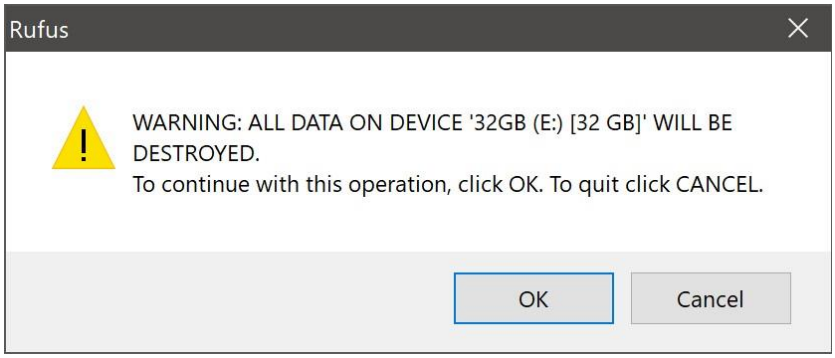
Write warnings

You will then be alerted that Rufus has detected that the Debian stable ISO is an ISOHybrid image. This means the same image file can be used as the source for both a DVD and a USB stick without requiring conversion.

Keep Write in ISO Image mode selected and click on OK to continue.



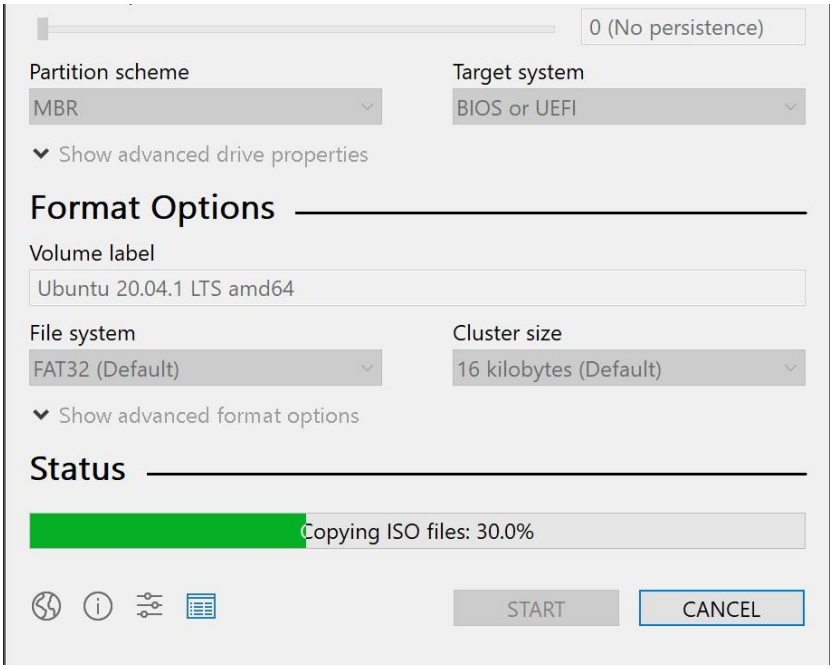
Rufus will also warn you that all data on your selected USB device is about to be destroyed. This is a good moment to double check you've selected the correct device before clicking OK when you're confident you have.



If your USB stick contains multiple partitions, Rufus will warn you in a separate pane that these will also be destroyed.

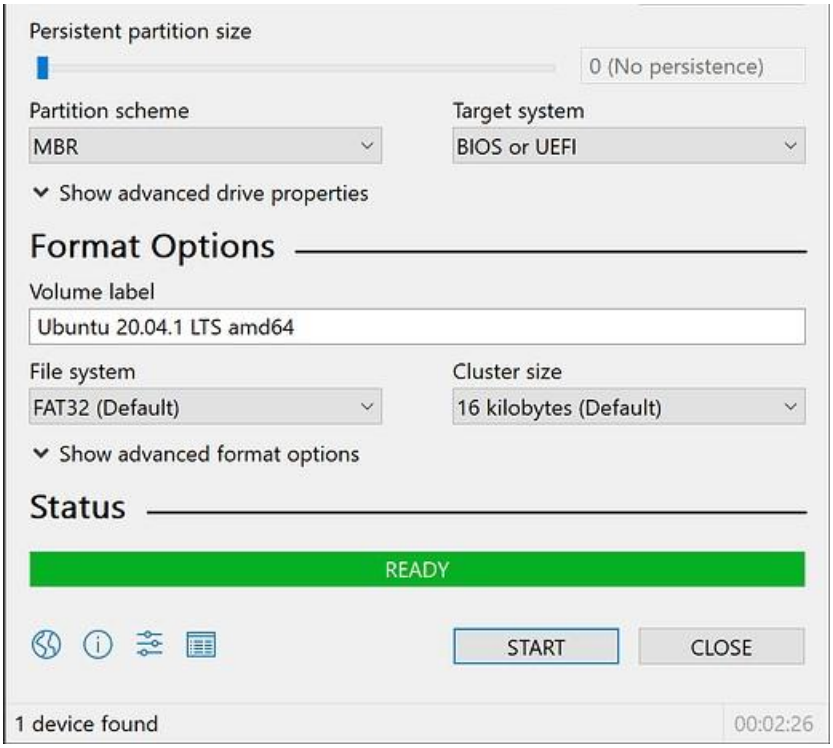
Writing the ISO

The ISO will now be written to your USB stick, and the progress bar in Rufus will give you some indication of where you are in the process. With a reasonably modern machine, this should take around 10 minutes. Total elapsed time is shown in the lower right corner of the Rufus window.



Installation complete

When Rufus has finished writing the USB device, the Status bar will be filled green and the word READY will appear in the center. Select CLOSE to complete the write process.



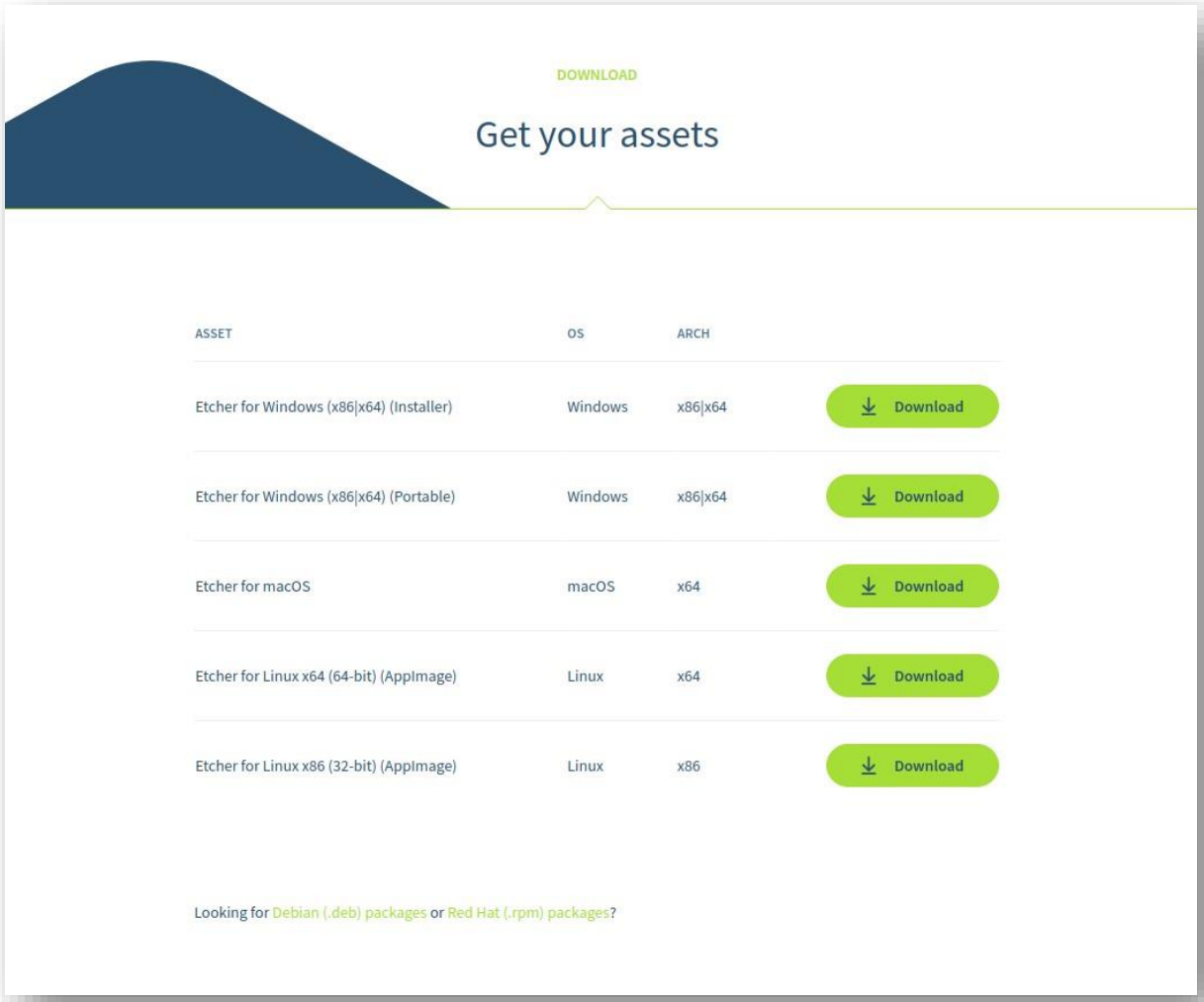
Congratulations! You now have Debian stable on a USB stick, bootable and ready to go.

To use it you need to insert the stick into your target PC or laptop and reboot the device. It should recognise the installation media automatically during startup but you may need to hold down a specific key (usually F12) to bring up the boot menu and choose to boot from USB.

Create a Bootable USB stick by using Balena etcher

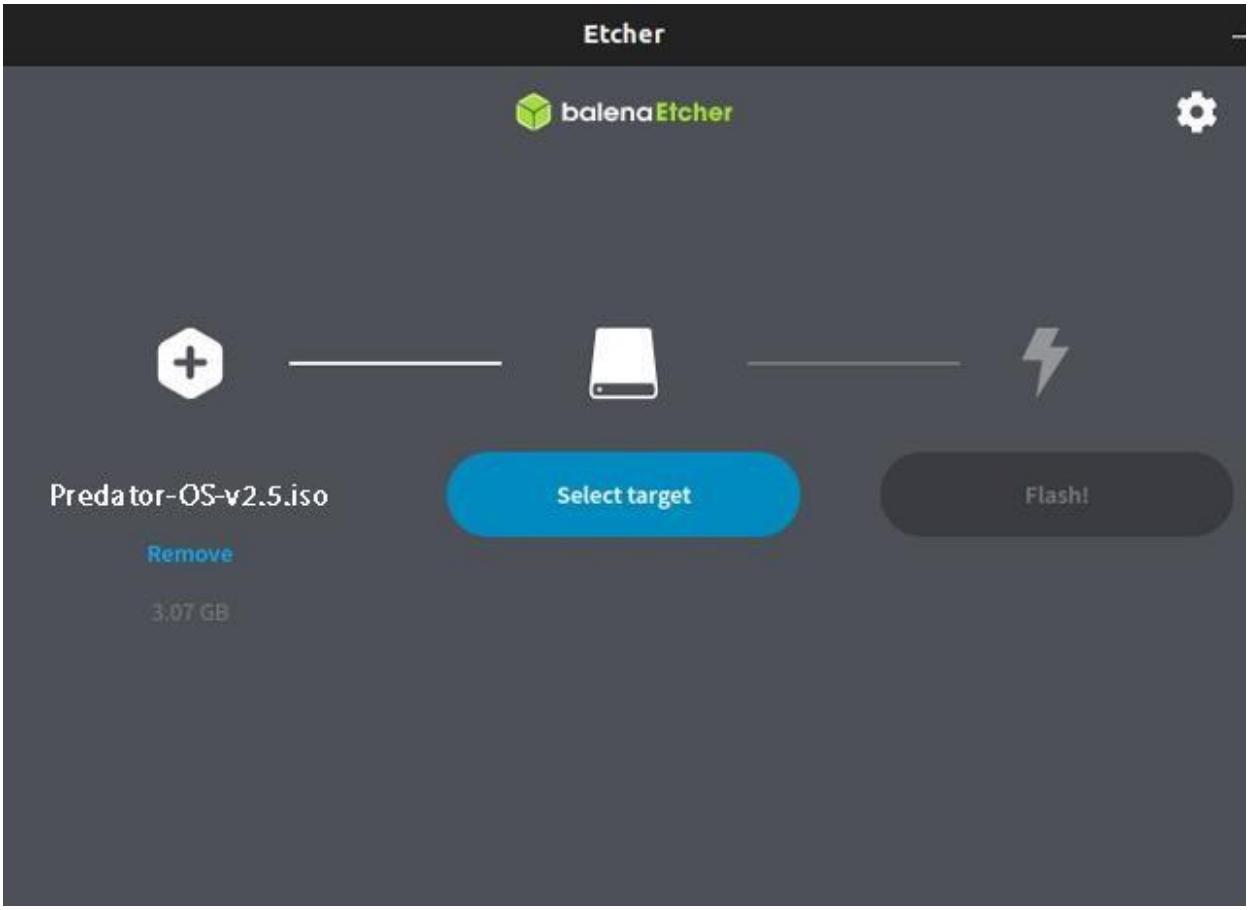
To install Predator-OS Linux, you need to write your downloaded ISO to a USB stick to create the installation media. This is not the same as copying the ISO, and requires some bespoke software. For this tutorial, we will use balenaEtcher, as it runs on Linux, Windows and Mac OS. Choose the version that corresponds to your current operating system, download and install the tool.

https://www.balena.io/etcher/



If you prefer to use a different tool to create your USB, we also have tutorials for Rufus on Windows, Etcher on Mac OS and Startup Disk Creator on Debian stable.

Select your downloaded ISO, choose your USB flash drive, and then click Flash! to install your image.



Create a Bootable USB stick by using Disk Creator

Launch Startup Disk Creator

We're going to use an application called 'Startup Disk Creator' to write the ISO image to your USB stick. This is installed by default on Debian stable, and can be launched as follows:

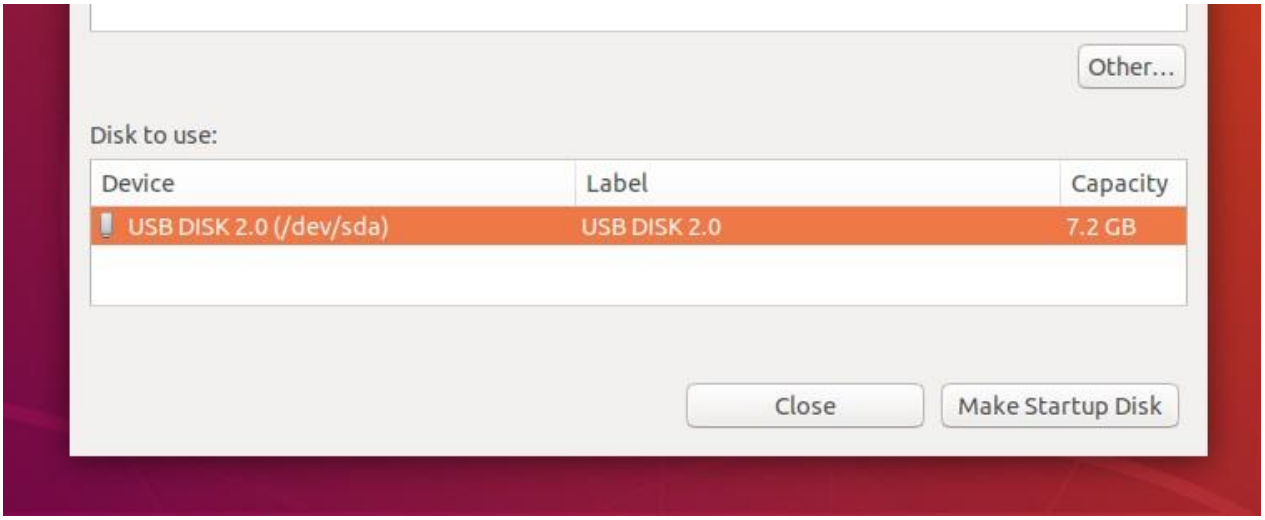
1. Insert your USB stick (select 'Do nothing' if prompted by Debian stable)
2. On Debian stable 18.04 and later, use the bottom left icon to open 'Show Applications'
3. In older versions of Debian stable, use the top left icon to open the dash
4. Use the search field to look for Startup Disk Creator
5. Select Startup Disk Creator from the results to launch the application

ISO and USB selection

When launched, Startup Disk Creator will look for the ISO files in your Downloads folder, as well as any attached USB storage it can write to.

It's likely that both your Predator-OS ISO and the correct USB device will have been detected and set as 'Source disc image' and 'Disk to use' in the application window. If not, use the 'Other' button to locate your ISO file and select the exact USB device you want to use from the list of devices.

Click Make Startup Disk to start the process.



Confirm USB device

Before making any permanent changes, you will be asked to confirm the USB device you've chosen is correct. This is important because any data currently stored on this device will be destroyed.

After confirming, the write process will start and a progress bar appears.

Installation complete

That's it! You now have Debian stable on a USB stick, bootable and ready to go.
If you want to install Debian stable, take a look at our install Debian stable desktop tutorial.

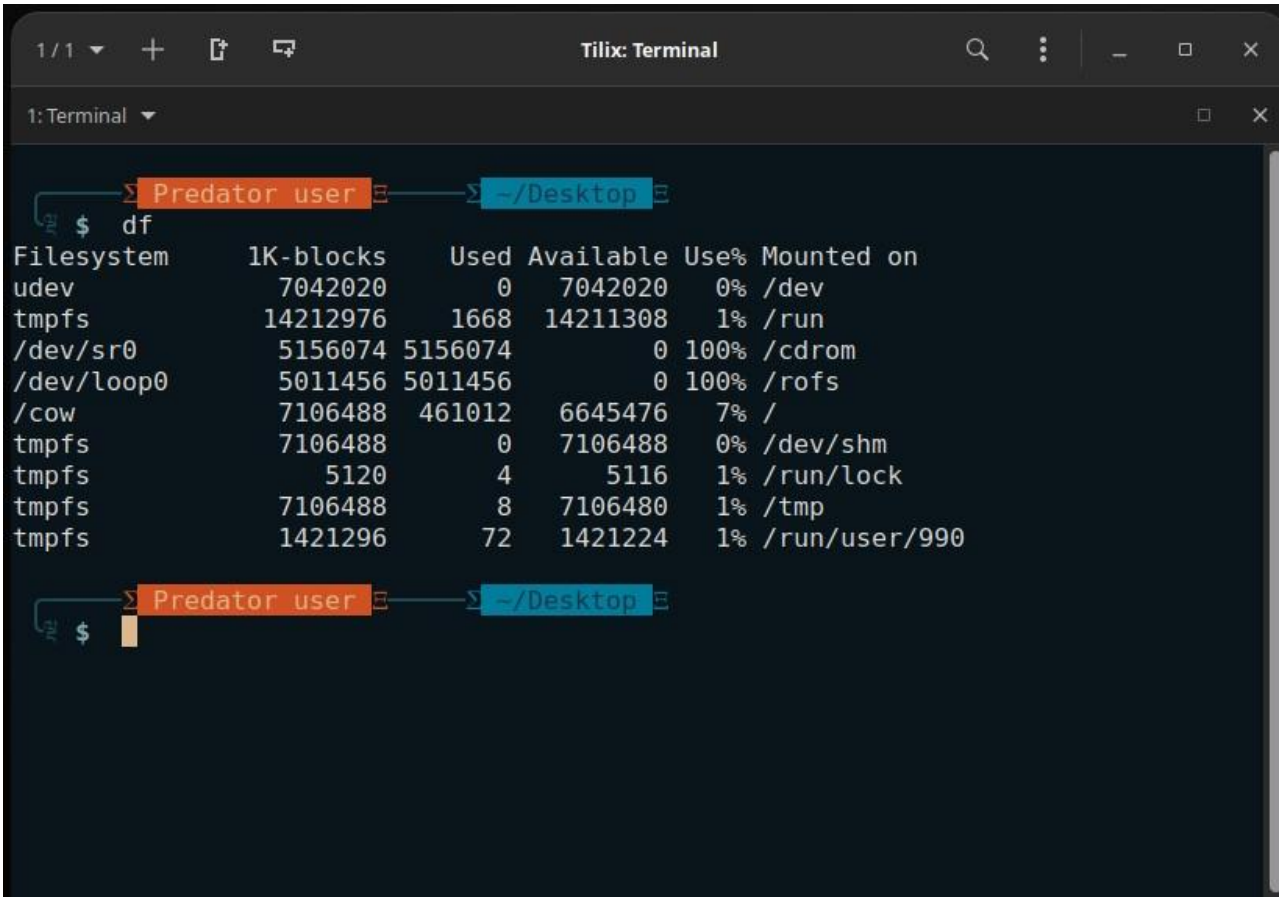
Create a bootable USB device on Linux using dd command

Step 1 – Download Predator-OS .iso image

Step 2 – Find your usb device name on Linux

Insert your USB stick and type the following df command to see if it is mounted automatically on a Debian or any other Linux desktop system:

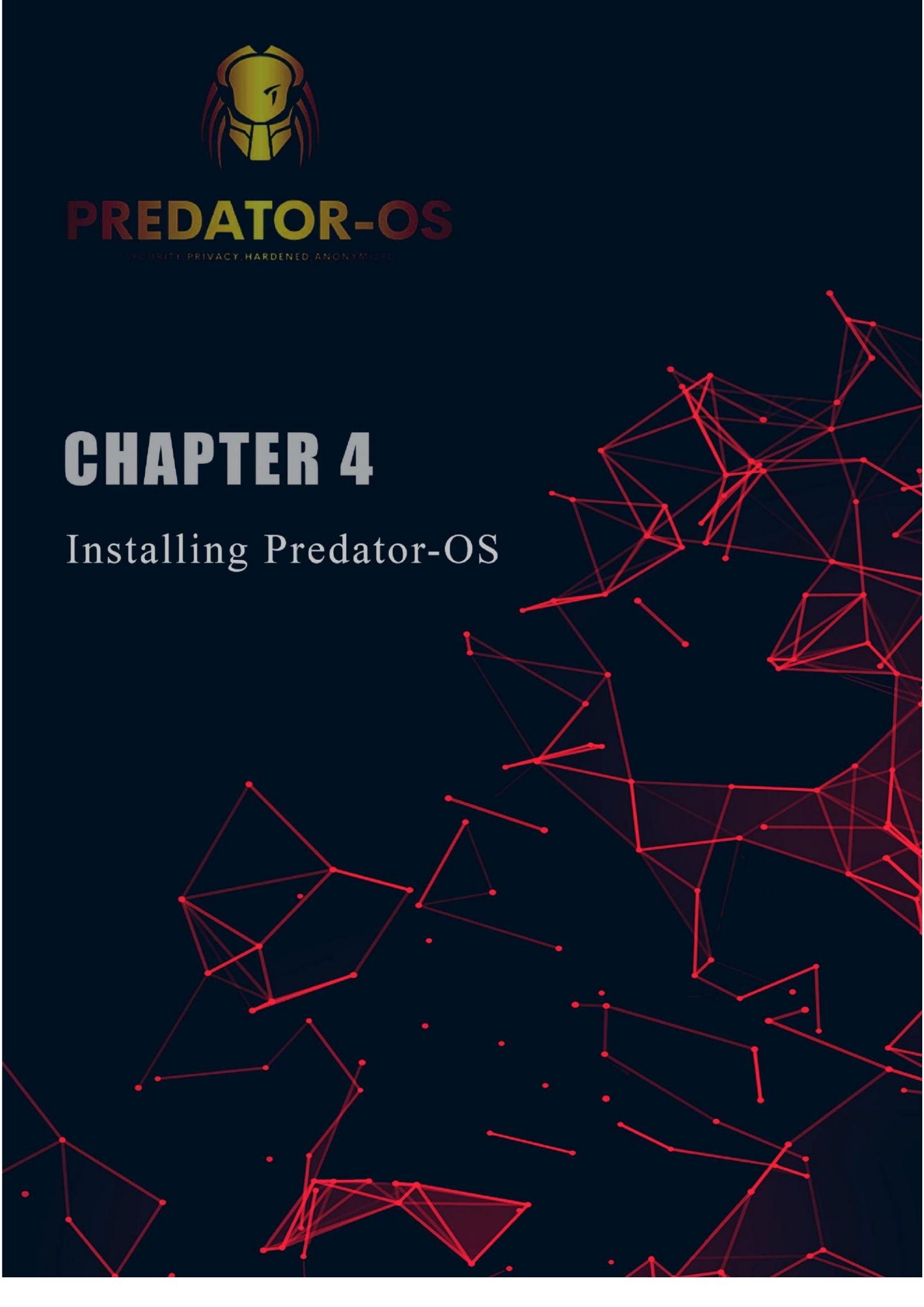
```
$ df
```



Step 4 – Create a bootable USB stick on Linux

Another example:

$ sudo dd if= Predator-OS-3.1-LTS.iso f=/dev/sdd bs=1M status=progres



CHAPTER 4

Installing Predator-OS

Chapter 4

Installing Predator-OS

Installing the Predator-OS Linux on a real computer

Boot from USB flash drive

Insert the USB flash drive into the laptop or PC you want to use to install Debian stable and boot or restart the device. It should recognise the installation media automatically. If not try holding F12 during startup and selecting the USB device from the system-specific boot menu.

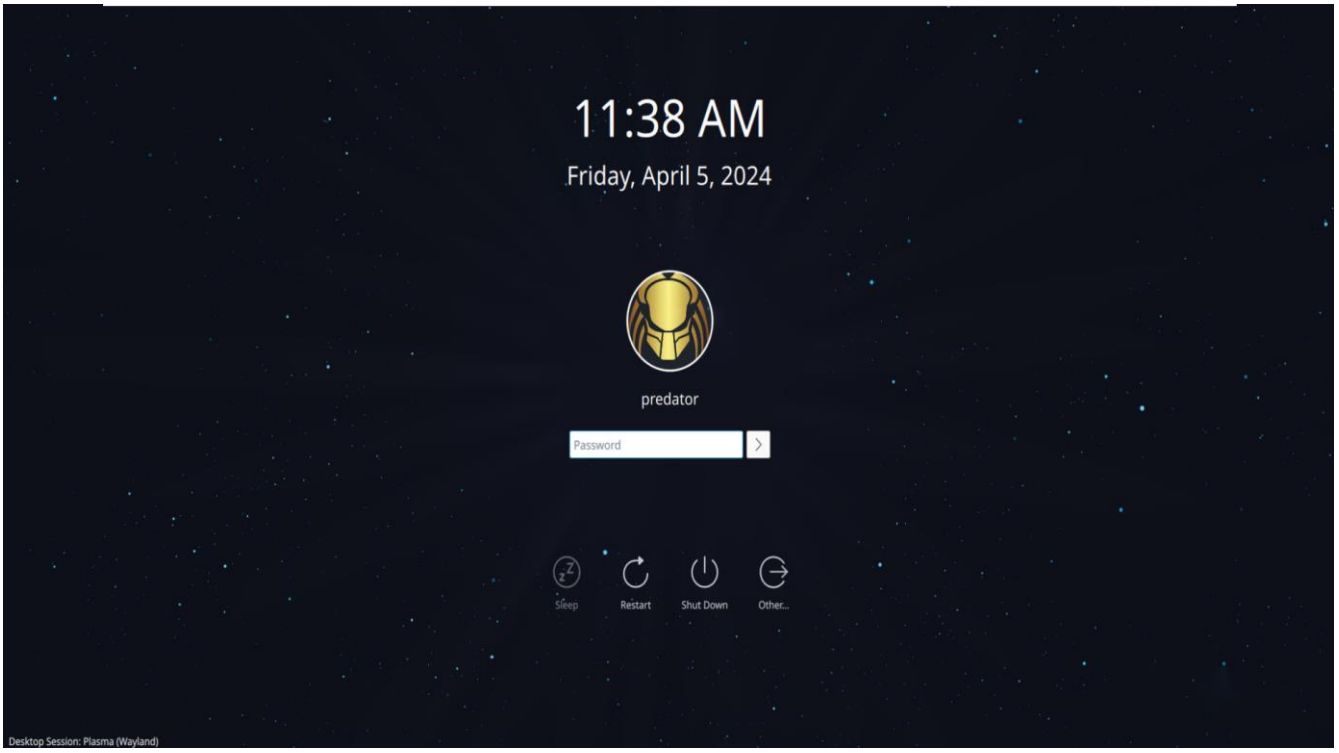
F12 is the most common key for bringing up your system's boot menu, but Escape, F2 and F10 are common alternatives. If you're unsure, look for a brief message when your system starts – this will often inform you of which key to press to bring up the boot menu. You should now see the welcome screen inviting you to either try or install Debian stable.

Complete the Installation

Now sit back and enjoy the slideshow as Predator-OS installs in the background! Once the installation has completed, you will be prompted to restart your machine. Click Restart Now.

When you restart, you will be prompted to remove your USB flash drive from the device. Once you've done this press ENTER.
Enter your password on the login screen (assuming you selected that option when creating your login details).

The user that has already been created can then log in and begin working immediately.



And that's it. Welcome to your new Predator-OS Desktop!

PREDATOR-OS

SECURITY, PRIVACY, HARDENED, ANONYMIZED

CHAPTER 5

Installation in virtual machines

Chapter 5

Installation in virtual machines
In a Virtual Box

Introduction

VirtualBox allows you to run an entire operating system inside another operating system. Please be aware that you should have a minimum of 2 GB of RAM. 4 GB of RAM or more is recommended.

Comparison to Dual-Boot

Many websites (including the one you're reading) have tutorials on setting up dual-boots between Windows and Predator-OS. A dual-boot allows you; at boot time, to decide which operating system you want to use. Installing Predator-OS on a virtual machine inside of Windows has a lot advantages over a dual-boot (but also a few disadvantages).

Advantages of virtual installation

The size of the installation does not have to be predetermined. It can be a dynamically resized virtual hard drive. You do not need to reboot in order to switch between Predator-OS and Windows.The virtual machine will use your Windows internet connection so you don't have to worry about Predator-OS not detecting your wireless card if you have one. The virtual machine will set up its own video configuration, so you don't have to worry about installing proprietary graphics drivers to get a reasonable screen resolution. You always have Windows to fall back on in case there are any problems. All you have to do is press the right Control key instead of rebooting your entire computer.

For troubleshooting purposes, you can easily take screenshots of any part of Predator-OS (including the boot menu or the login screen). It's low commitment. If you later decide you don't like Predator-OS, all you have to do is delete the virtual hard drive and uninstall VirtualBox.

Disadvantages of virtual installation

In order to get any kind of decent performance, you need at least 512 MB of RAM, because you are running an entire operating system (Predator-OS) inside another entire operating system (Windows). The more memory the better. I would recommend at least 1 GB of RAM.

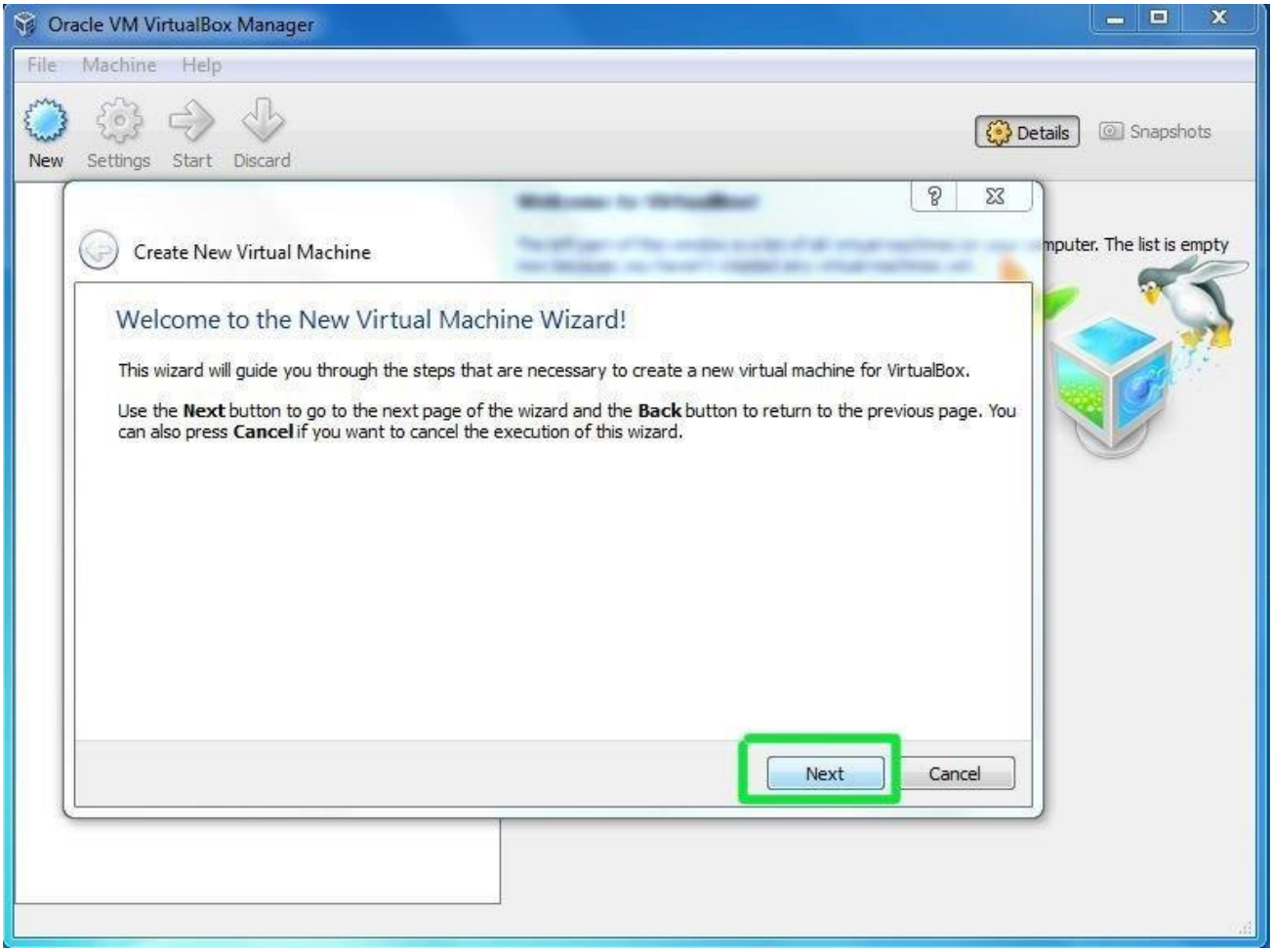
Even though the low commitment factor can seem like an advantage at first, if you later decide you want to switch to Predator-OS and ditch Windows completely you

cannot simply delete your Windows partition. You would have to find some way to migrate out your settings from the virtual machine and then install Predator-OS over Windows outside the virtual machine.
Every time you want to use Predator-OS, you have to wait for two boot times (the time it takes to boot Windows, and then the time it takes to boot Predator-OS within Windows).

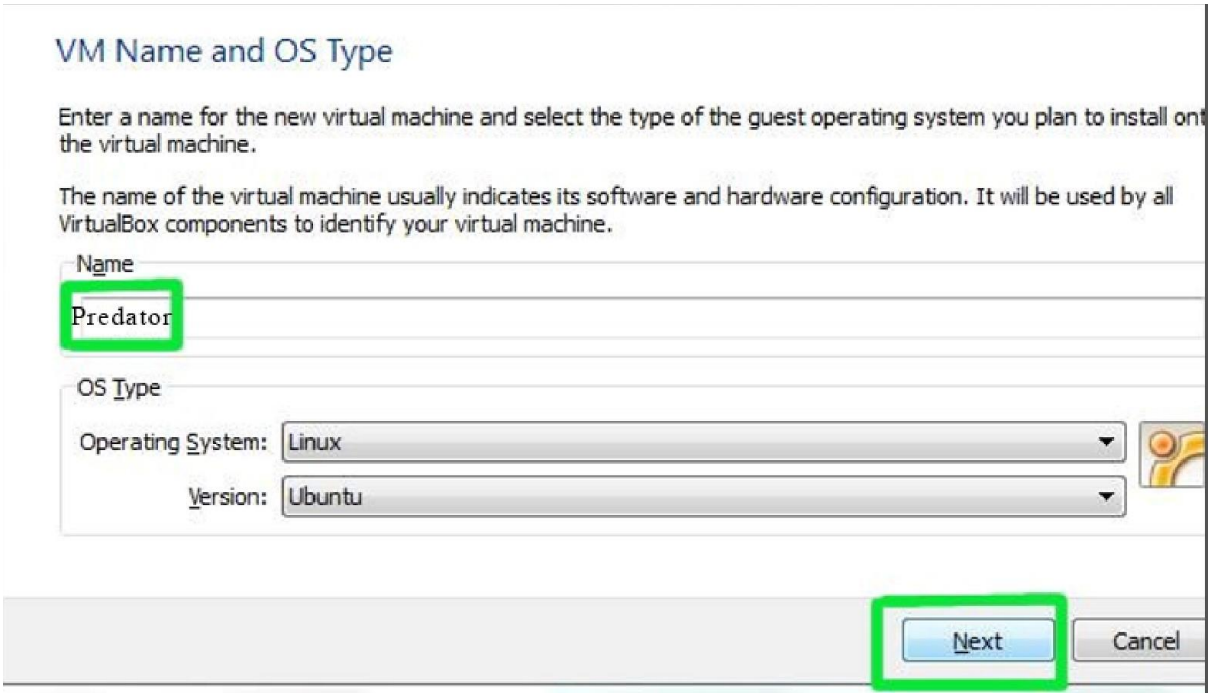
Installation Process

Follow these instructions to get a Predator-OS disk image (.iso file).

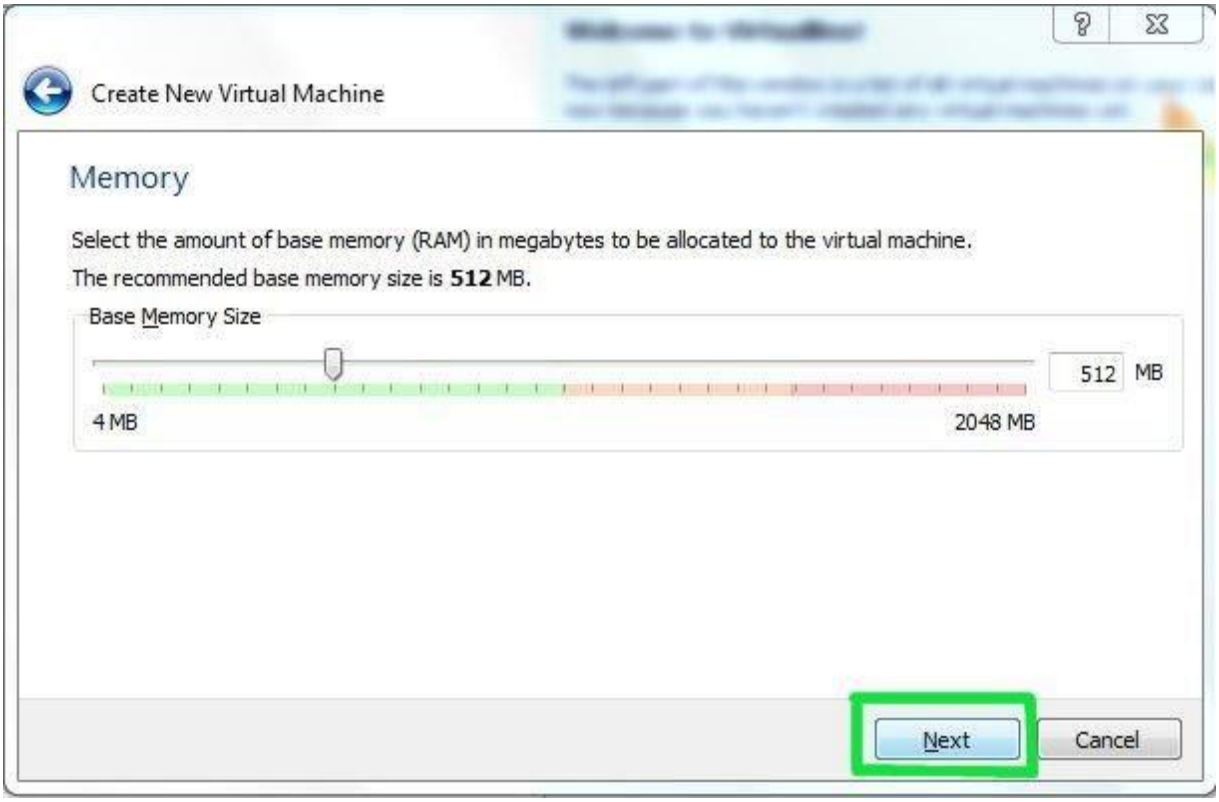
Steps to Install Predator-OS On VirtualBox



After you launch VirtualBox from the Windows Start menu, click on **New** to create a new virtual machine. When the New Virtual Machine Wizard appears, click **Next**.



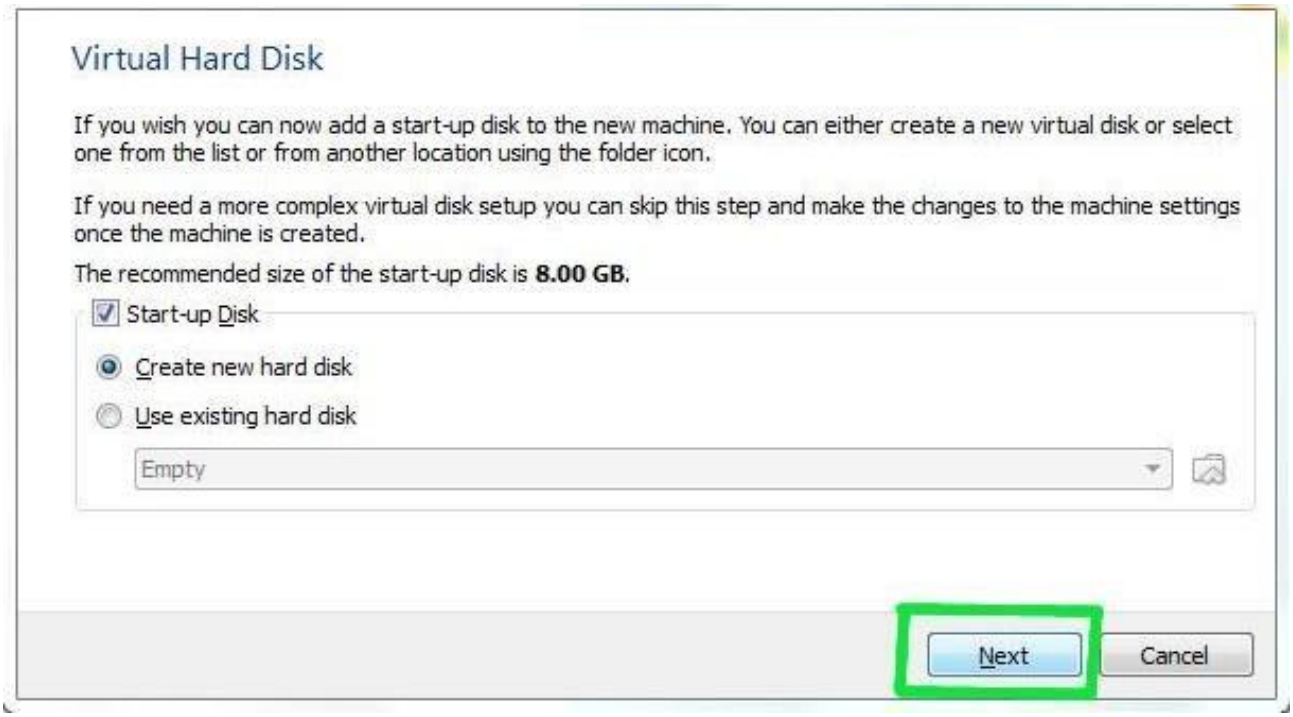
You can call the machine whatever you want. If you're installing Predator-OS, it makes sense to call it **Predator-OS**, I guess. You should also specify that the operating system is **Linux**.



VirtualBox will try to guess how much of your memory (or RAM) to allocate for the virtual machine. If you have 1 GB or less of RAM, I would advise you stick with the recommendation. If, however, you have over 1 GB, about a quarter your RAM or less should be fine. For example, if you have 2 GB of RAM, 512 MB is

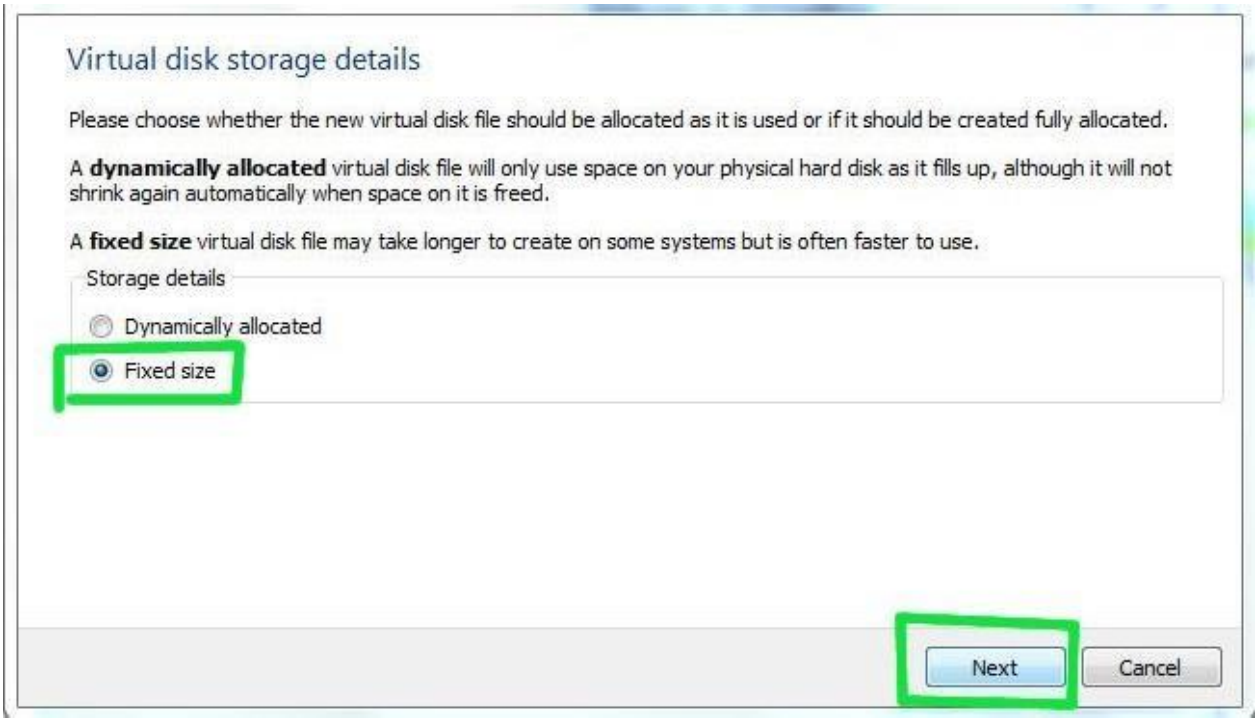
fine to allocate. If you have 4 GB of RAM, 1 GB is fine to allocate. If you have no idea what RAM is or how much of it you have, just go with the default.

Click **Next**.

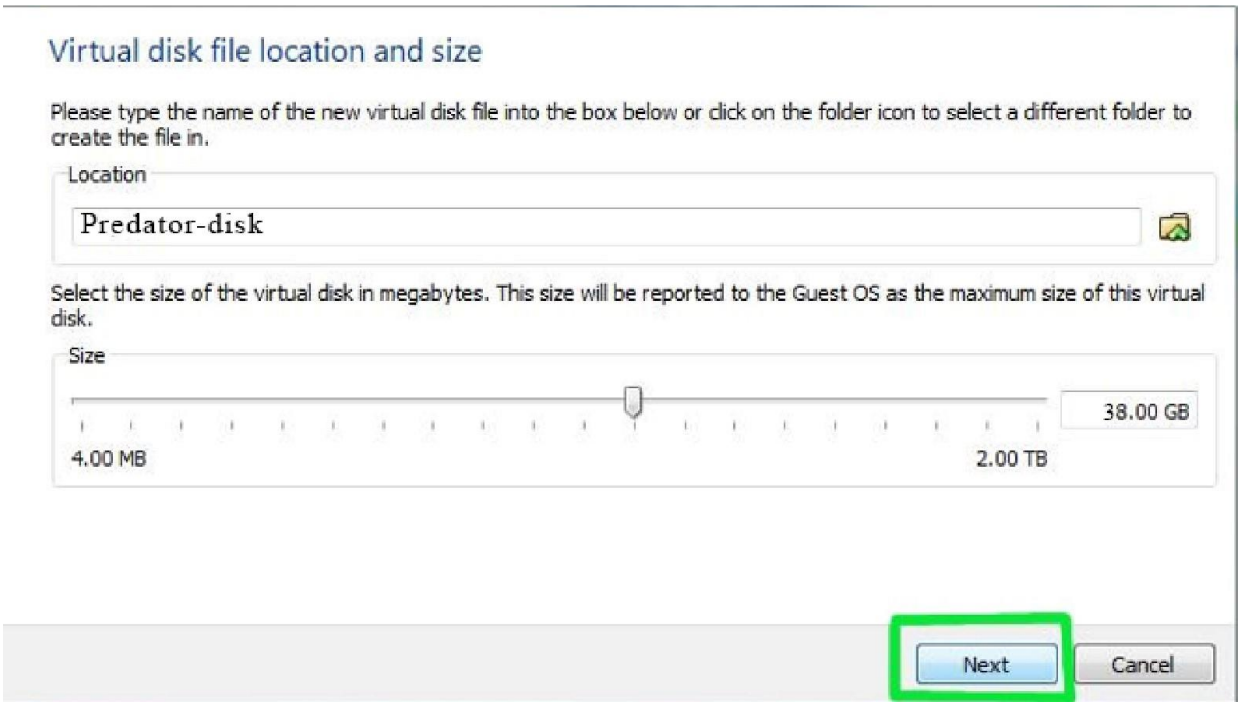


If this is your first time using VirtualBox (which it probably is if you need a tutorial on how to use it), then you do want to *Create new hard disk* recommended 30 GB disk space and then click **Next**.

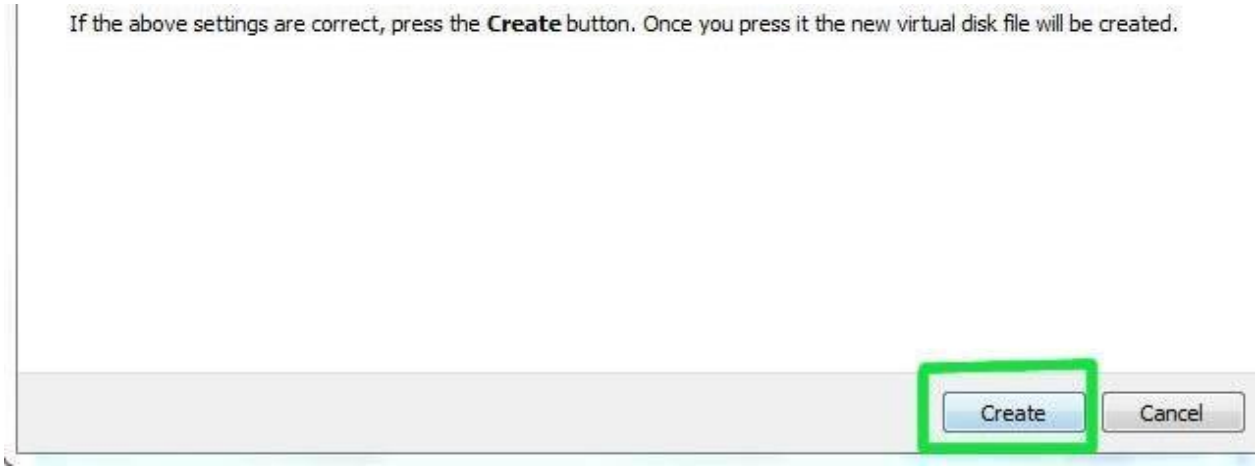
Click **Next** again.



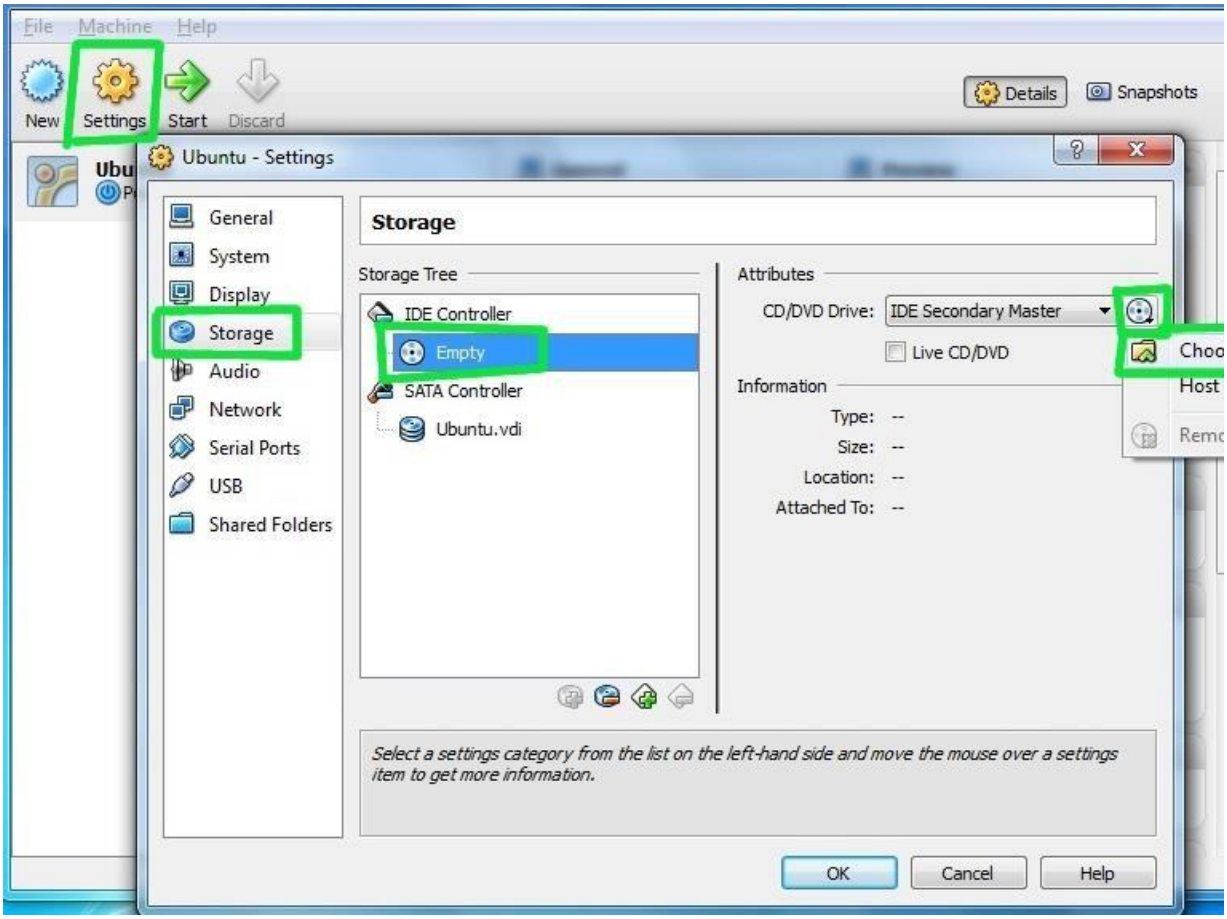
Theoretically a dynamically expanding virtual hard drive is best because it'll take up only what you actually use. I have come upon weird situations though, when installing new software in a virtualized Predator-OS, in which the virtual hard drive just fills up instead of expanding. So I would actually recommend picking a **Fixed-size storage**.



Predator-OS's default installation is less than 25 GB. If you plan on adding software or downloading large files in your virtualized Predator-OS, you should tack on some buffer.

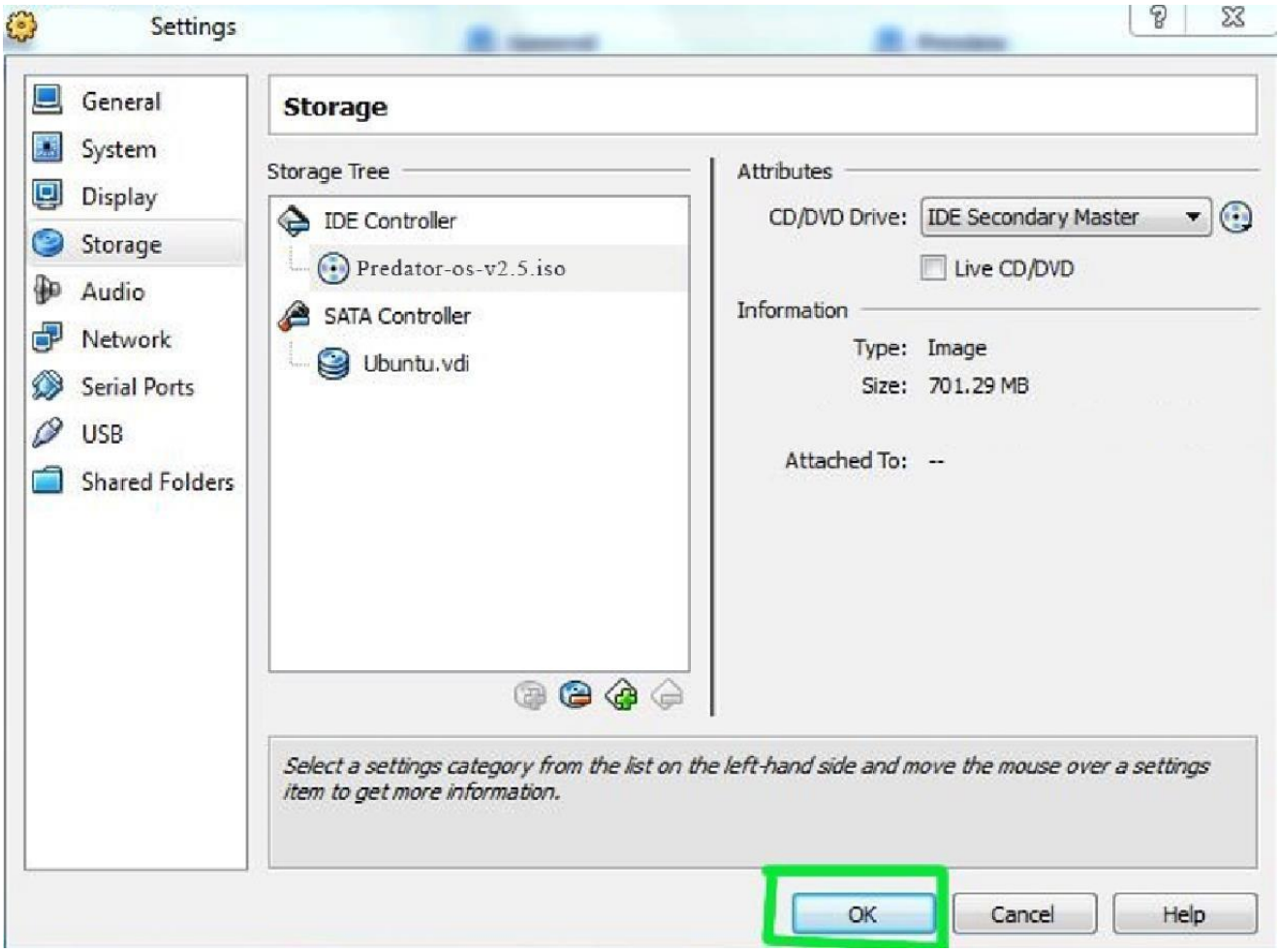


Click **Create** and wait for the virtual hard drive to be created. This is actually just a very large file that lives inside of your Windows installation.



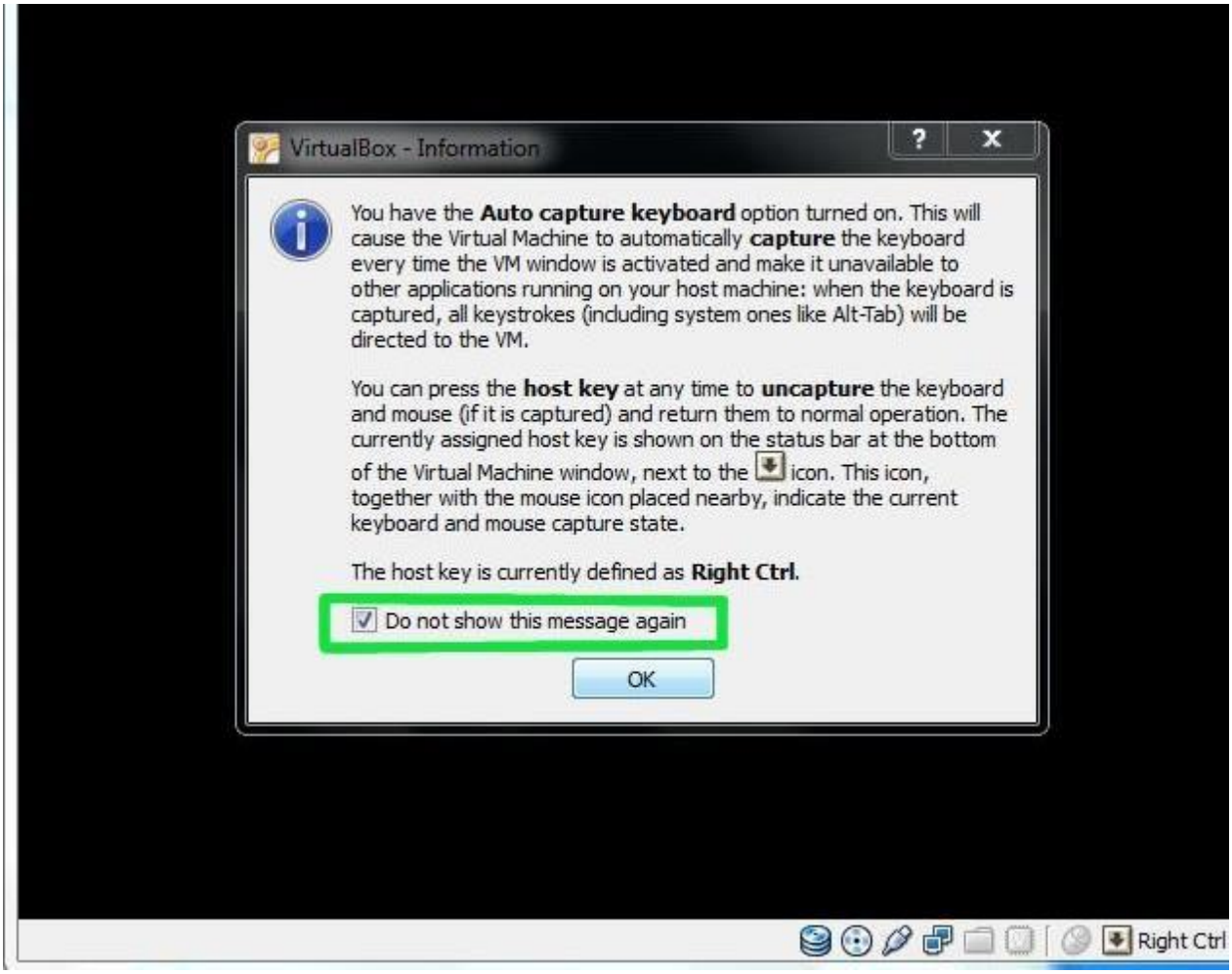
The next thing to do to make the (currently blank) virtual hard drive useful is to add the downloaded Predator-OS disk image (the .iso) boot on your virtual machine. Click on **Settings** and **Storage**. Then, under *CD/DVD Device*, next to *Empty*, you will see a little folder icon. Click that.

Select the Predator-OS .iso you downloaded earlier.



Once you've selected it, click **OK**.

Then double-click your virtual machine to start it up.



You may get a bunch of random warnings/instructions about how to operate the guest operating system within VirtualBox. Read those, and then you may also want to mark not to see those again.

The installation of the Predator-OS will be beginning now.

In VMware workstation

As more as the technology develops it gives more flexibility, usability, and portability. With the introduction of cross platform technology like virtualization and remote computing, any program can be run on any platform irrespective of compatibility. Hardware virtualizations allows to create a fully working virtual machine on any platform. VMWare Workstation is one such hardware visualization platform available in the market on which any virtual machines can be created.

What Is VMWare Workstation?

VMWare Workstation is an application developed by VMWare to create virtual machines, containers, and Kubernetes clusters on any desktop or server system. VMWare released VMWare Workstation in two products: VMWare Workstation Pro and VMWare Workstation Player. VMWare Workstation Player is released on a free license with limited features, whereas VMWare Workstation Pro is an enterprise paid version that has loaded with a lot of features. Full documentation is published on the VMWare portal for free. Refer the documentation from here.

Can I install Predator-OS in Vmware Workstation on other Windows versions?

Yes, you can install Predator-OS in Vmware Workstation on Windows 8, 8.1, and 7. In the past I have not only installed Predator-OS but also Windows XP, 7 and Fedora on my Windows system.

Virtualization is a good way to try Linux from the comfort of Windows. WSL and WSL2 might be easier but not everyone has access to them. And for a relatively better desktop experience, a VM is better.

You don't have to make actual changes to the disk partition, no changes in the boot and Linux runs like any other application inside Windows.

In this tutorial, I will show how to install Predator-OS inside Windows using VMWare.

Installing Linux inside Windows using VMWare

Your actual operating system is called host OS and the operating system you install in the virtual machine is called guest OS. I'll use this terminology in the tutorial here.

The virtual machines use your host OS's system resources. Predator-OS Plasma requires 4 GB of RAM to function properly, your system should have 8 GB to allocate 4 GB to the guest OS (Predator-OS) and keep 4 GB for the host OS (Windows).

Let's see all the requirements.

Requirements

- Good internet connection to download software and Linux ISO. (You can also use some other computer with an internet connection to download these files.)
- Windows system with at least 25 GB of free space. A 25GB+ Free space is good for installing the latest version of Predator-OS.
- Windows system with 8 GB of RAM. (It can work with less RAM as well, but your system will start to lag while using Linux in the virtual machine.)
- Make sure to enable virtualization in the BIOS

I am installing Predator-OS 22.04 in this tutorial, but the same steps apply to any other Linux distribution.

Step 1: Download and install VMWare Player

Go to VMWare website and download the .exe file of VMWare Player. At the time of writing this article, VMWare player 16 is the latest version.

Download VMWare

Once downloaded, double-click the exe file and follow the on-screen instructions to install VMWare.

Step 2: Download the Linux ISO

Next, you need to download the ISO file of the Linux distribution. You can get this image from the official website of the Linux distribution you are trying to use.

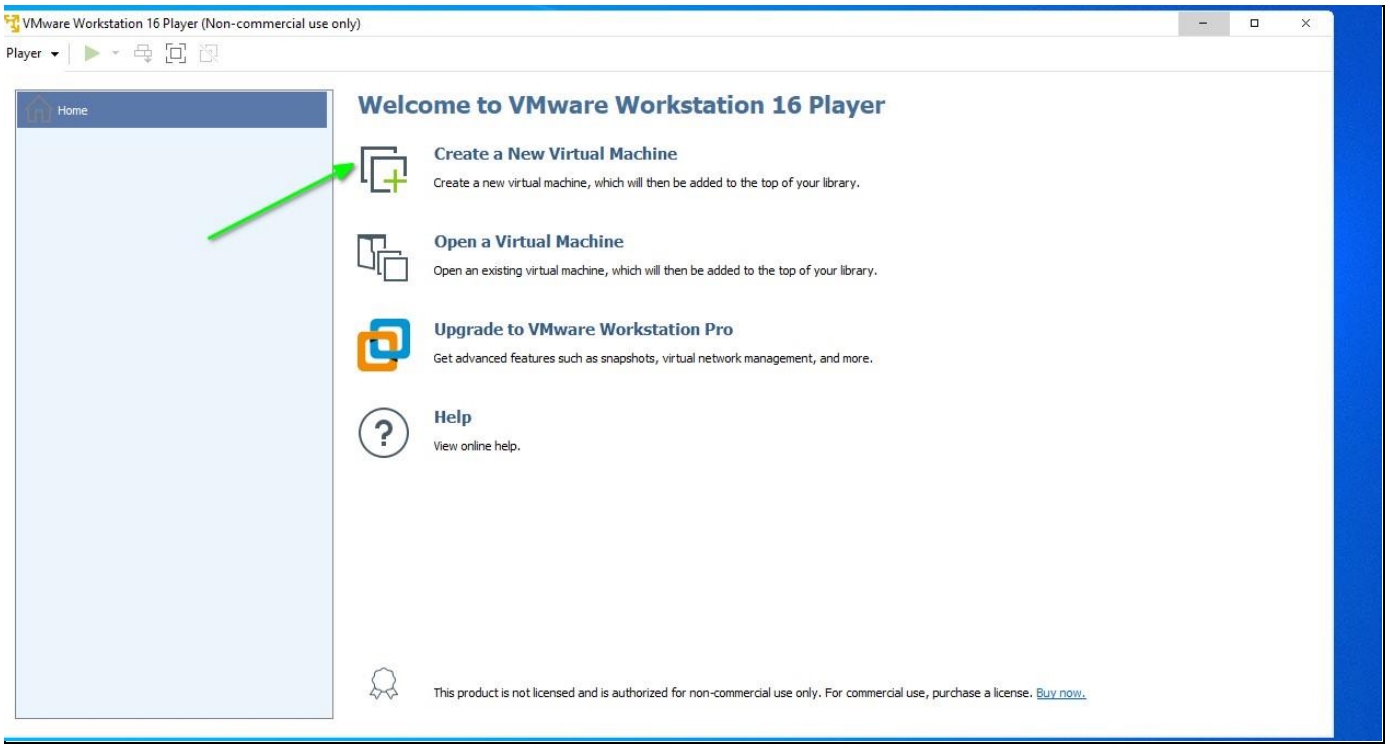
I am using Predator-OS in this example, and you can download ISO images for Predator-OS from the link below:

Download Predator-OS

Step 3: Install Linux using VMWare

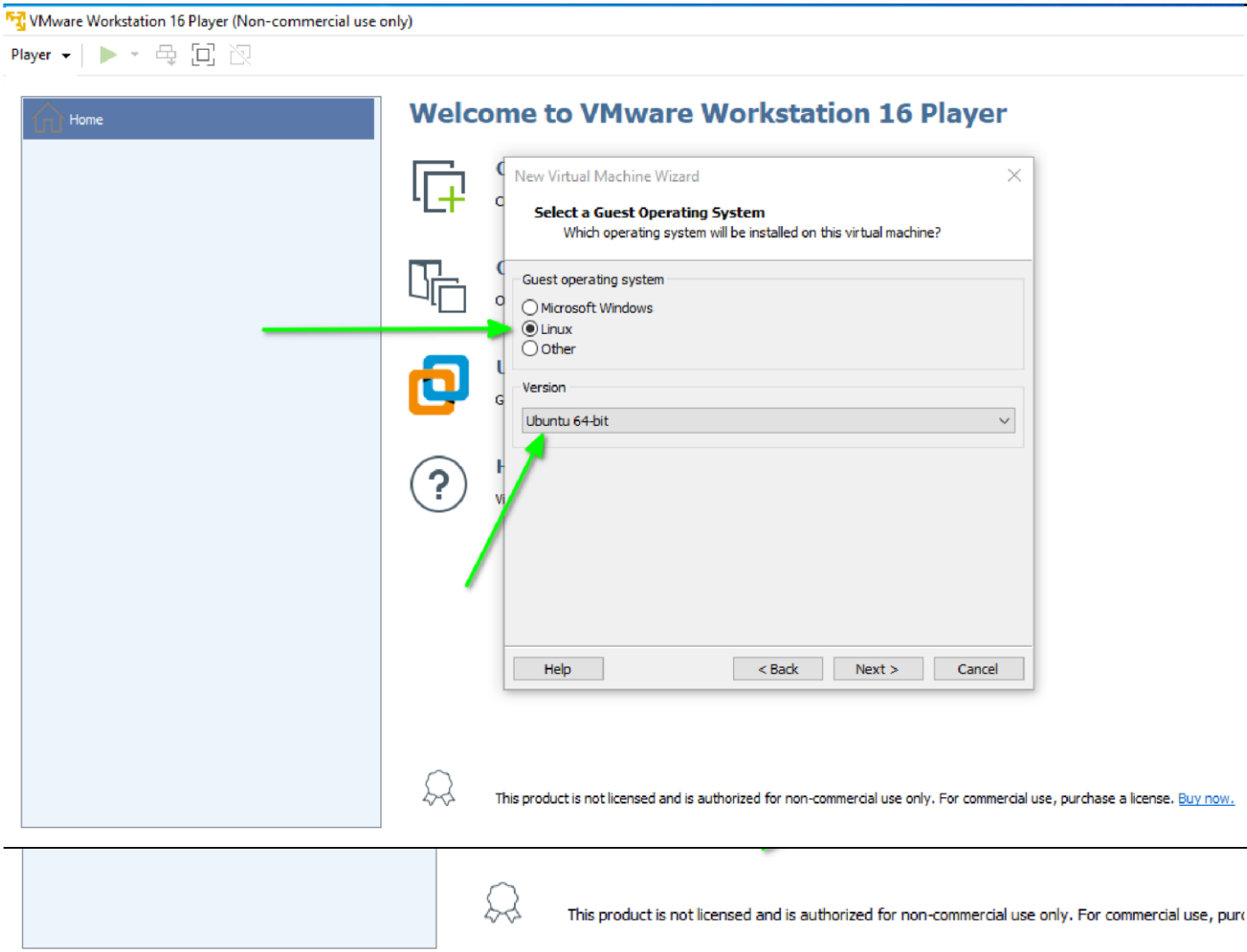
You have installed VMWare and you have downloaded the ISO for Linux. You are now set to install Linux in VMware.

Now, start VMWare and click on **Create New Virtual Machine**.



Create new virtual machine in VMWare

Select “I will install operating system later” option and press next.

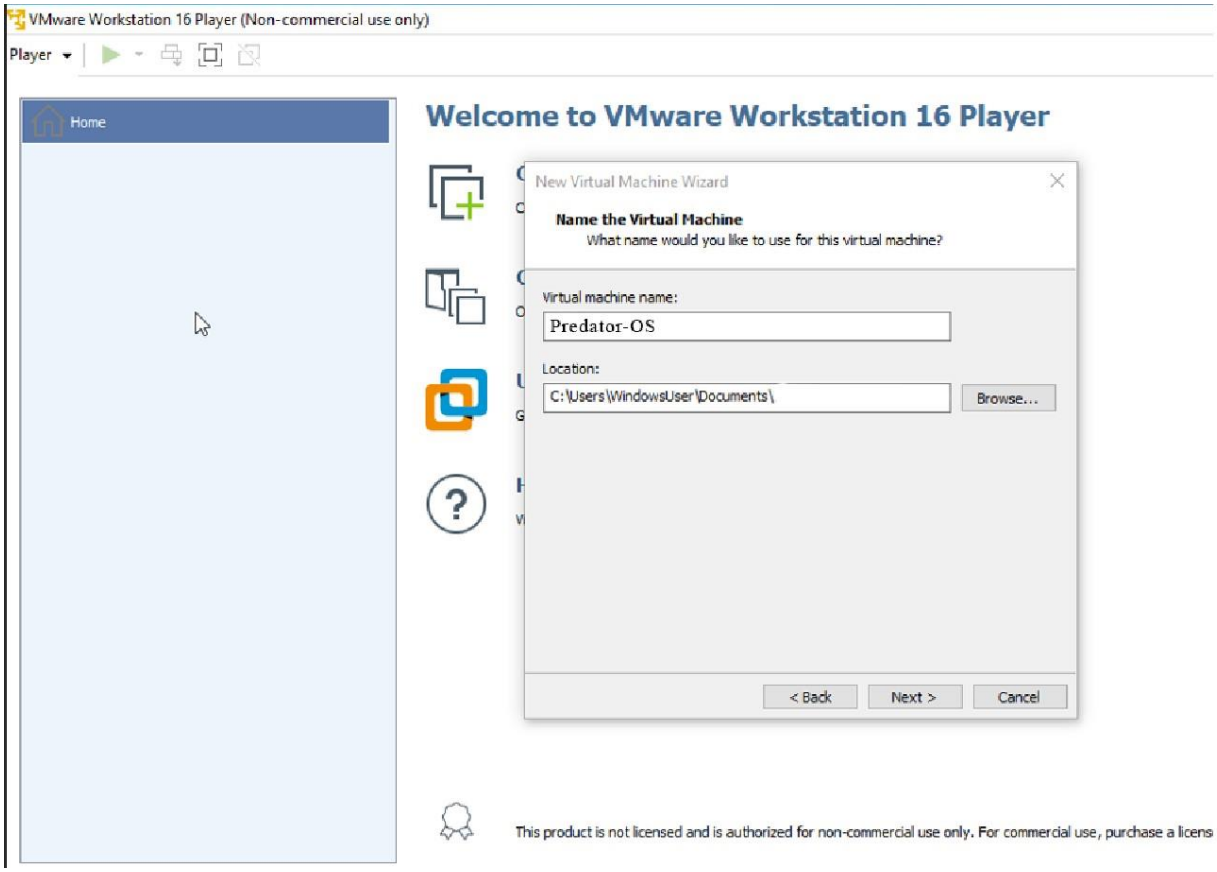


Select install operating system later button

On the next screen, set the Operating system to Linux and the version to Predator-OS 64bit.

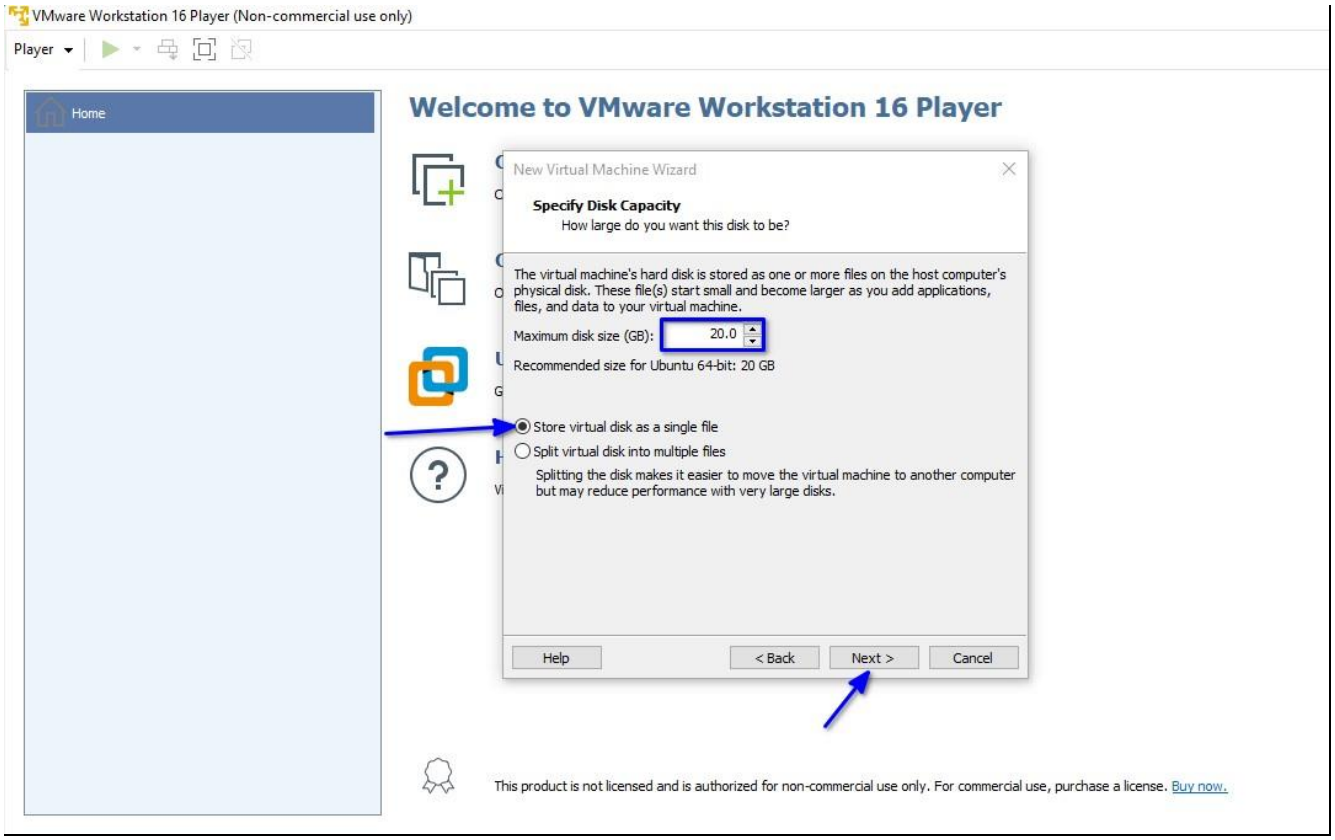
Select Linux type and Predator-OS 64 type

Give the virtual machine a name and press Next.



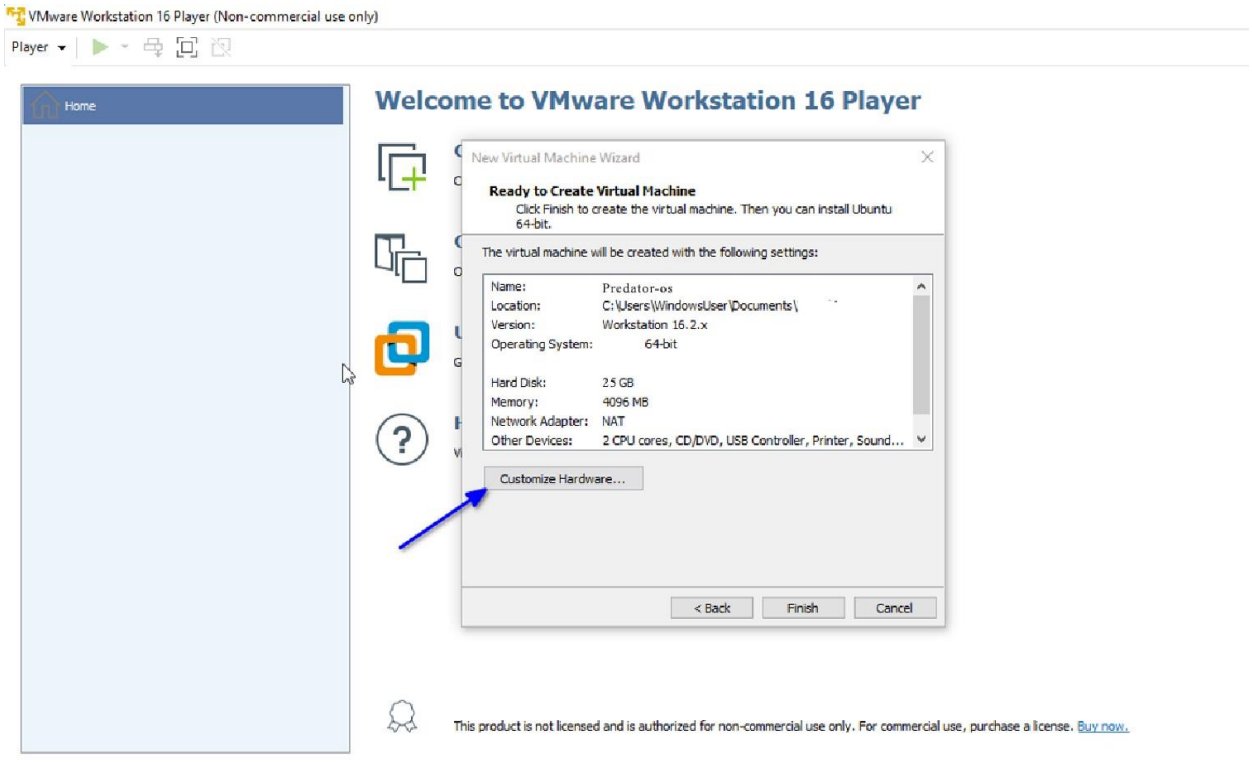
Name the virtual machine

In the next screen, set the disk size to a minimum of 25 GB and also select "Store Virtual Disk as a single file" option.



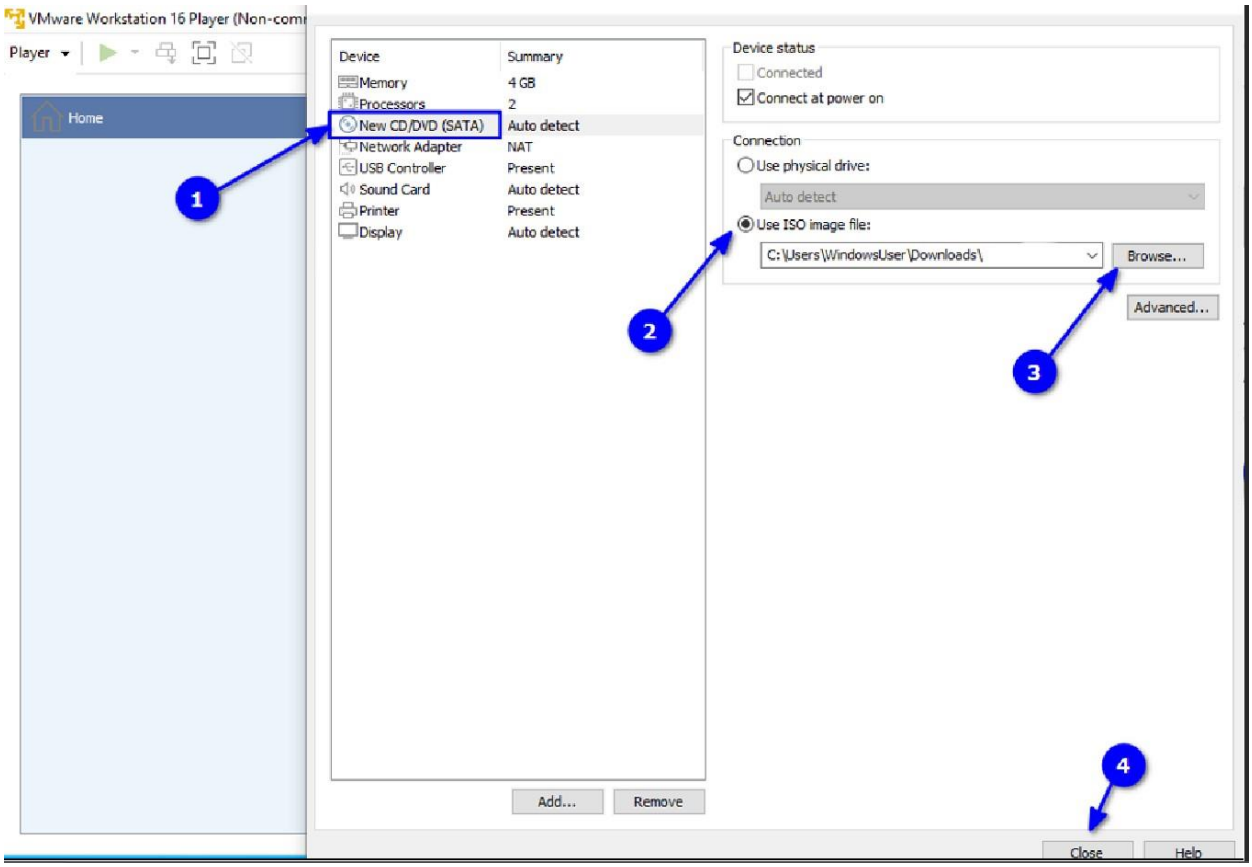
Select disk size and store as single file

From the next screen, you can either press Finish and set ISO file later by right clicking and Settings. Or you can select the ISO file on the go. For this, press "Customize Hardware" button.



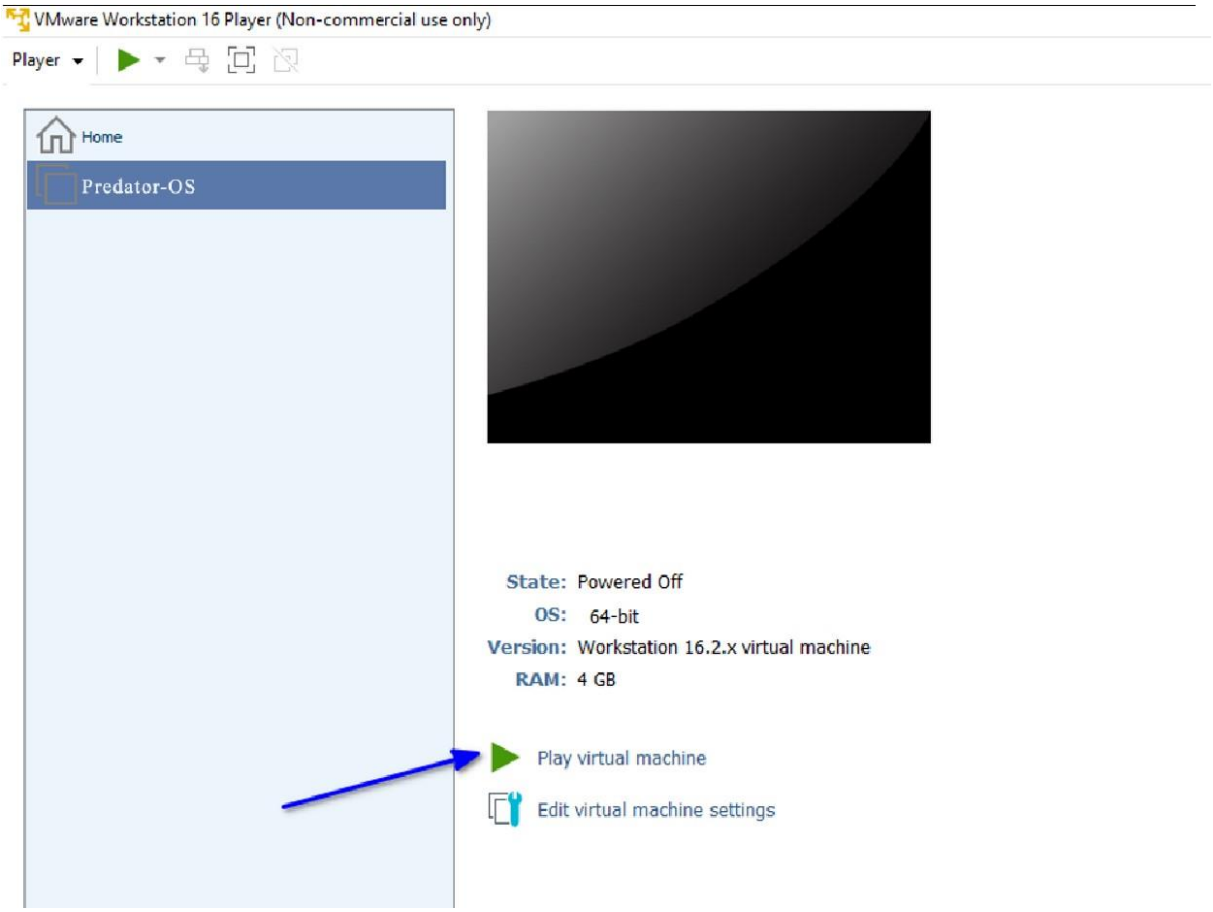
Press Customise Hardware button

On this screen, you can tweak memory, processors, etc. But you need to select "New CD/DVD" button and add the Predator-OS ISO as shown in the screenshot:



Set ISO image file from customize hardware dialog box

Now, you can close this and press the finish button. Once done, you can now start the VMWare virtual machine and start the installation of Predator-OS.

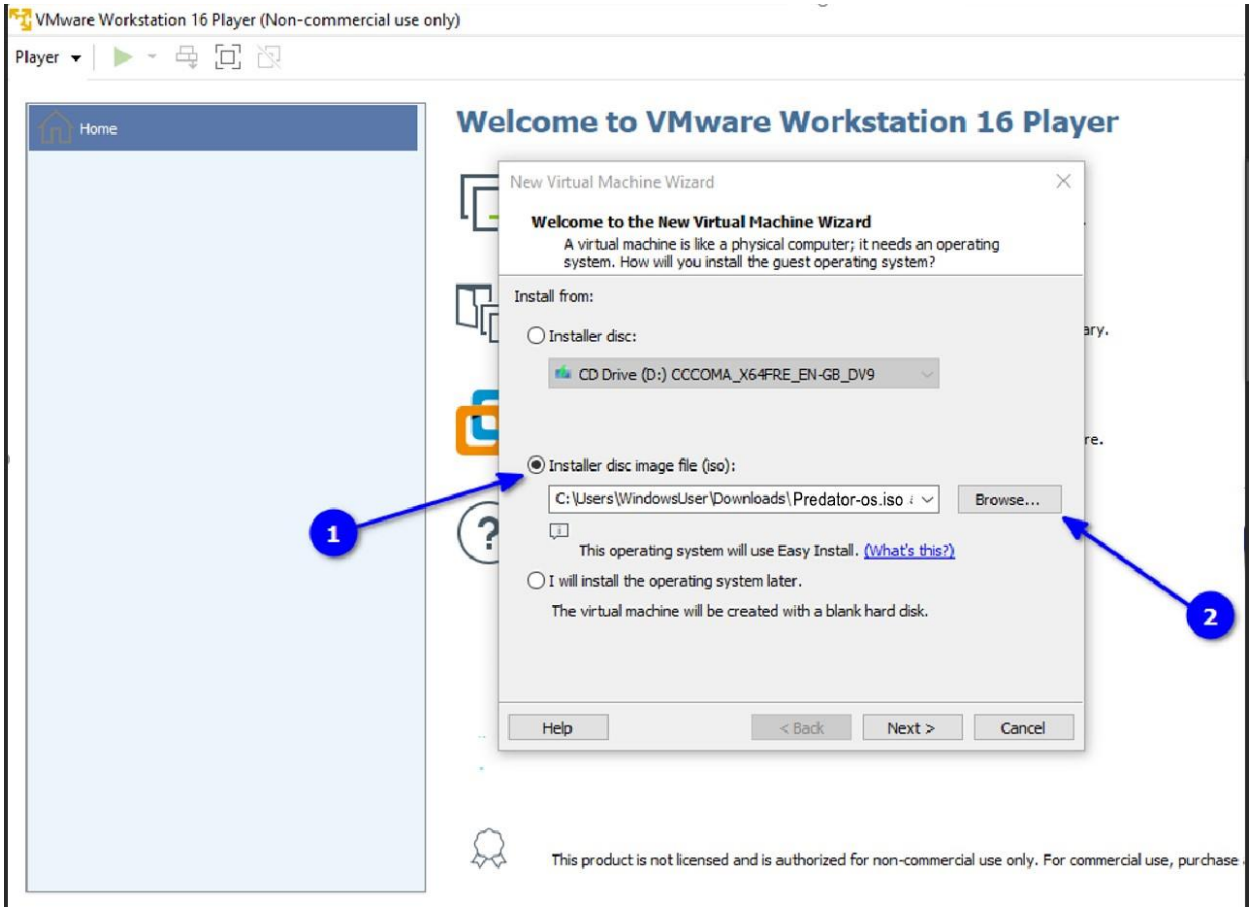


Select the virtual machine to start install

Now, you can use it after installation by opening the virtual machine by name from VMWare home screen.

Quick note about VMWare Easy install

This is another mode of installation in VMWare. Here, instead of pressing "ISO file later," you can select the downloaded ISO file and start the process.



Select disc install option for easy install

But I found both the process almost the same because the installer asked for the same steps in the previous method here also.

Installing Predator-OS in VMWare will be beginning now.

How to Install QEMU/KVM on Predator-OS to Create Virtual Machines

Virtualization is one of the most widely used technologies both in enterprise and home environments. Whether you are a seasoned IT expert, a programmer, or an IT novice, virtualization can be one of your greatest friends.

Virtualization is the abstraction of a computer's hardware resources using a software application known as a hypervisor. The hypervisor creates an abstraction layer over computer hardware and virtualizes various components of the system including but not limited to memory, processor, storage, USB devices, etc.

In doing so, it allows you to create virtual computers also known as virtual machines off of the virtualized elements, and each virtual machine; also known as a guest, runs independently from the host system.

KVM, short for Kernel-based Virtual Machine is an open-source type 1 hypervisor (bare metal hypervisor) that is integrated into the Linux kernel. It allows you to create and manage virtual machines running Windows, Linux, or UNIX variants such as FreeBSD, and OpenBSD.
As mentioned earlier, each virtual machine has its own virtual resources such as storage, memory, CPU, network interfaces, USB interfaces, and video graphics to mention a few.

QEMU (**Quick Emulator**) is a software module that emulates various components of computer hardware. It supports full virtualizations and works alongside **KVM** to provide a holistic virtualization experience.
In this guide, we will demonstrate how to install **QEMU/KVM** on **Predator-OS 20.04 / 22.04** distributions.

Step 1: Check Virtualization Enabled in Predator-OS

To start off check if your CPU supports virtualization technology. Your system needs to have an Intel VT-x (**vmx**) processor or AMD-V (**svm**) processor. To verify this, run the following egrep command.

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

If Virtualization is supported, the output should be greater than **0**, for example, 2,4,6, etc.

Alternatively, you can run the following grep command to display the type of processor your system supports. In our case, we are running Intel VT-x denoted by the **vmx** parameter.

```
$ grep -E --color '(vmx|svm)' /proc/cpuinfo
```

Enable Virtualization in Predator-OS

Equally important, check if **KVM** virtualization is supported by running the following command:

```
$ kvm-ok
```

Check KVM Virtualization in Predator-OS

If the **kvm-ok** utility is missing, install the **cpu-checker** package as follows.

```
$ sudo apt install cpu-checker -y
```

Now that we have verified that our system supports **KVM** virtualization, let us proceed and install **QEMU**.

Step 2: Install QEMU/KVM on Predator-OS 20.04/22.04

Next up, update the package lists and repositories as follows.

```
$ sudo apt update
```

Thereafter, install **QEMU/KVM** alongside other virtualization packages as follows:

```
$ sudo apt install qemu-kvm virt-manager virtinst libvirt-clients bridge-utils libvirt-daemon-system -y
```

Install Qemu on Predator-OS

Let us examine what role each of these packages plays.

- **qemu-kvm** – This is an open-source emulator that emulates the hardware resources of a computer.
- **virt-manager** – A Qt-based GUI interface for creating and managing virtual machines using the libvirt daemon.
- **virtinst** – A collection of command-line utilities for creating and making changes to virtual machines.
- **libvirt-clients** – APIs and client-side libraries for managing virtual machines from the command line.
- **bridge-utils** – A set of command-line tools for managing bridge devices.
- **libvirt-daemon-system** – Provides configuration files needed to run the virtualization service.

At this point, we have installed **QEMU** and all the essential virtualization packages. The next step is to start and enable the **libvirtd** virtualization daemon. So, run the following commands:

```
$ sudo systemctl enable --now libvirtd
$ sudo systemctl start libvirtd
```

Next, verify if the virtualization service is running as shown.

```
$ sudo systemctl status libvirtd
```

Start libvirtd Virtualization Service

From the output above, the **libvirtd** daemon is up and running as expected. Additionally, add the currently logged-in user to the **kvm** and **libvirt** groups as shown.

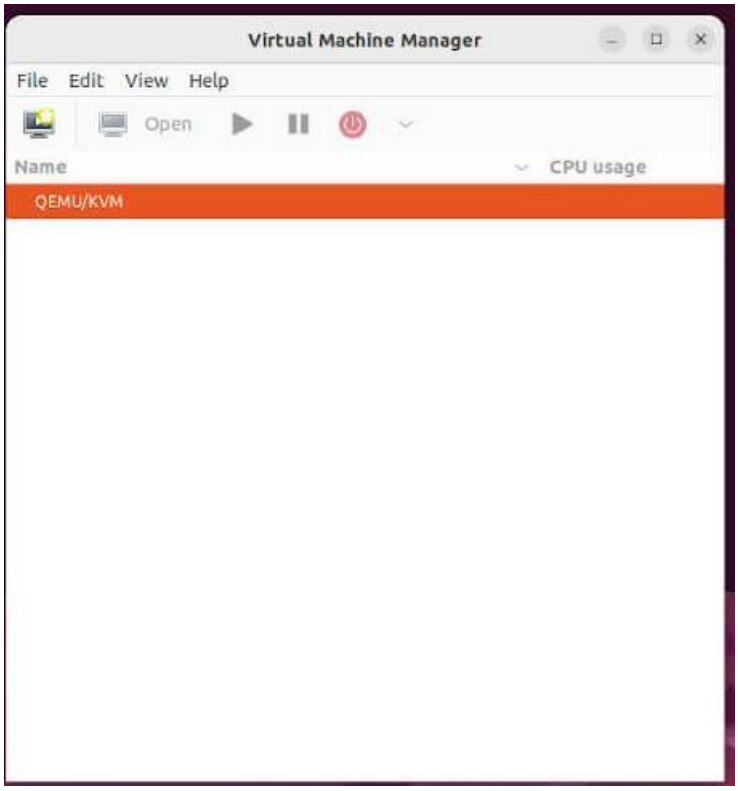
```
$ sudo usermod -aG kvm $USER
$ sudo usermod -aG libvirt $USER
```

Step 3: Launch Virtual Machine Manager in Predator-OS

The next step is to launch the **QEMU/KVM** GUI tool which is the **Virtual Machine Manager**.

```
$ sudo virt-manager
```

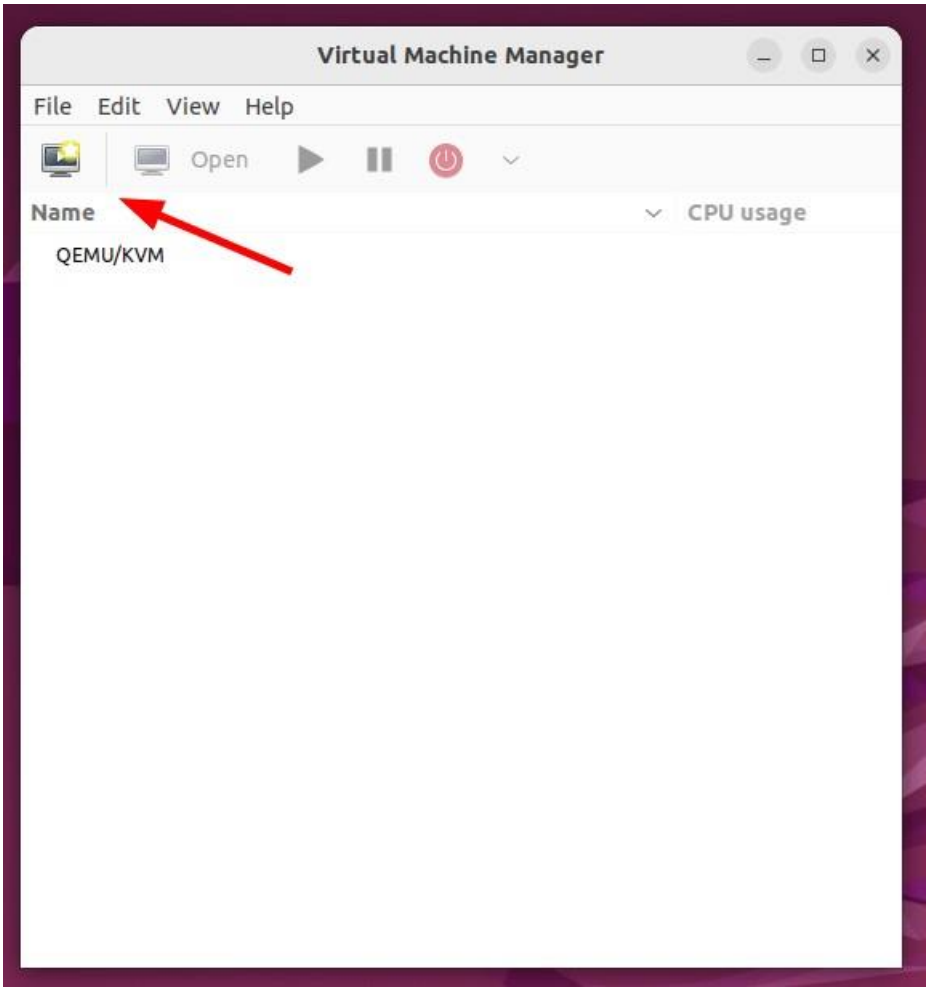
The **Virtual Machine Manager** will pop up as shown. From here, you can start creating and managing virtual machines as we shall demonstrate shortly.



Qemu Virtual Machine Manager

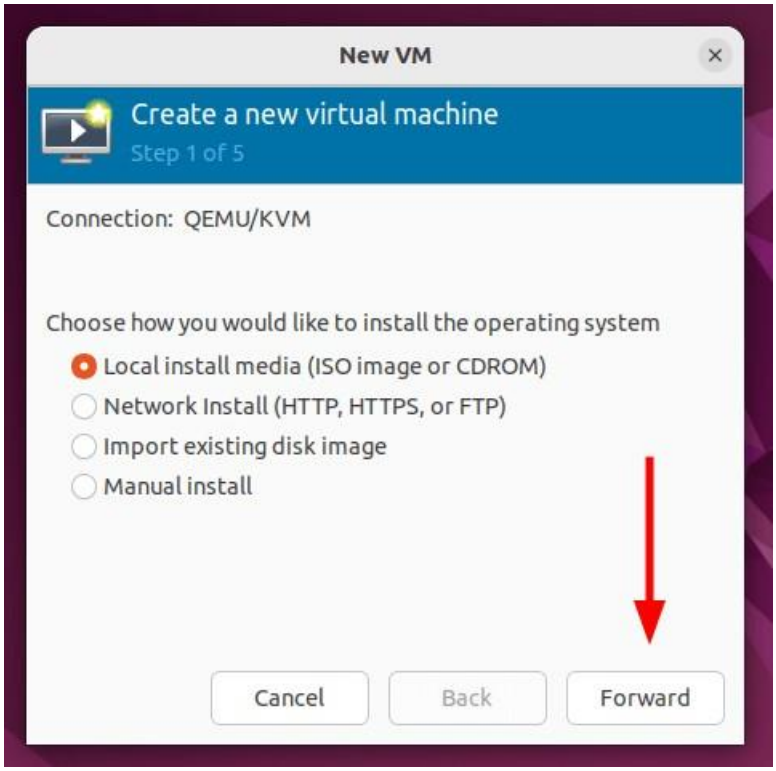
Step 4: Create Virtual Machine with QEMU/KVM in Predator-OS

In this section, we will demonstrate how you can create a virtual machine using an ISO image. You can use an ISO image of your preferred OS and follow along.
To begin, click on the icon at the top left corner as shown below.



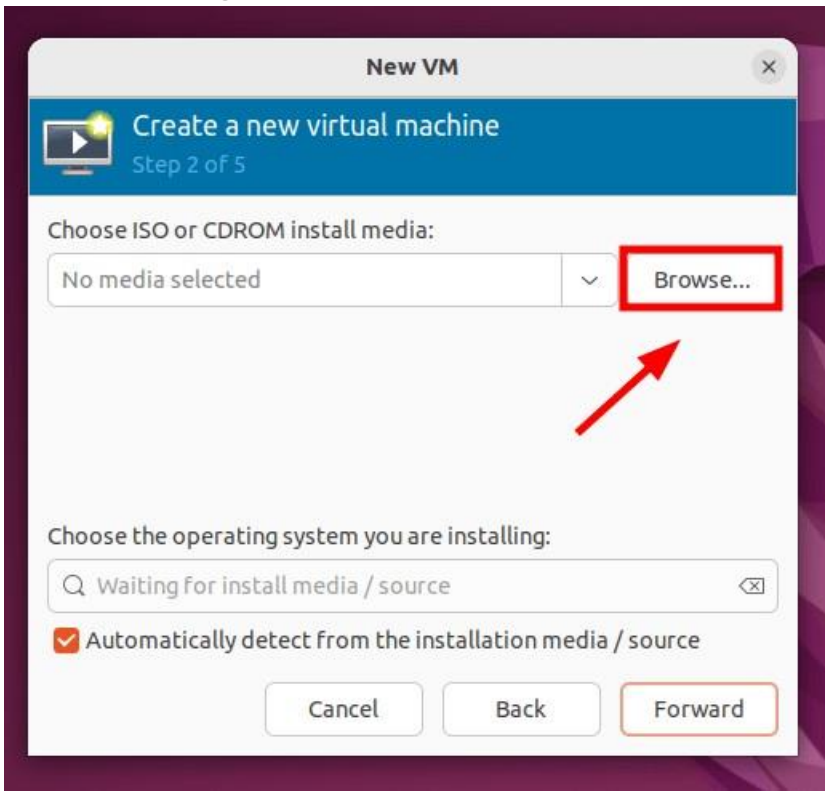
Create a Virtual Machine in Qemu

Since we are creating a virtual machine from an ISO file, select the first option – '**Local install media (ISO image or CDROM)**'. Then click '**Forward**'.



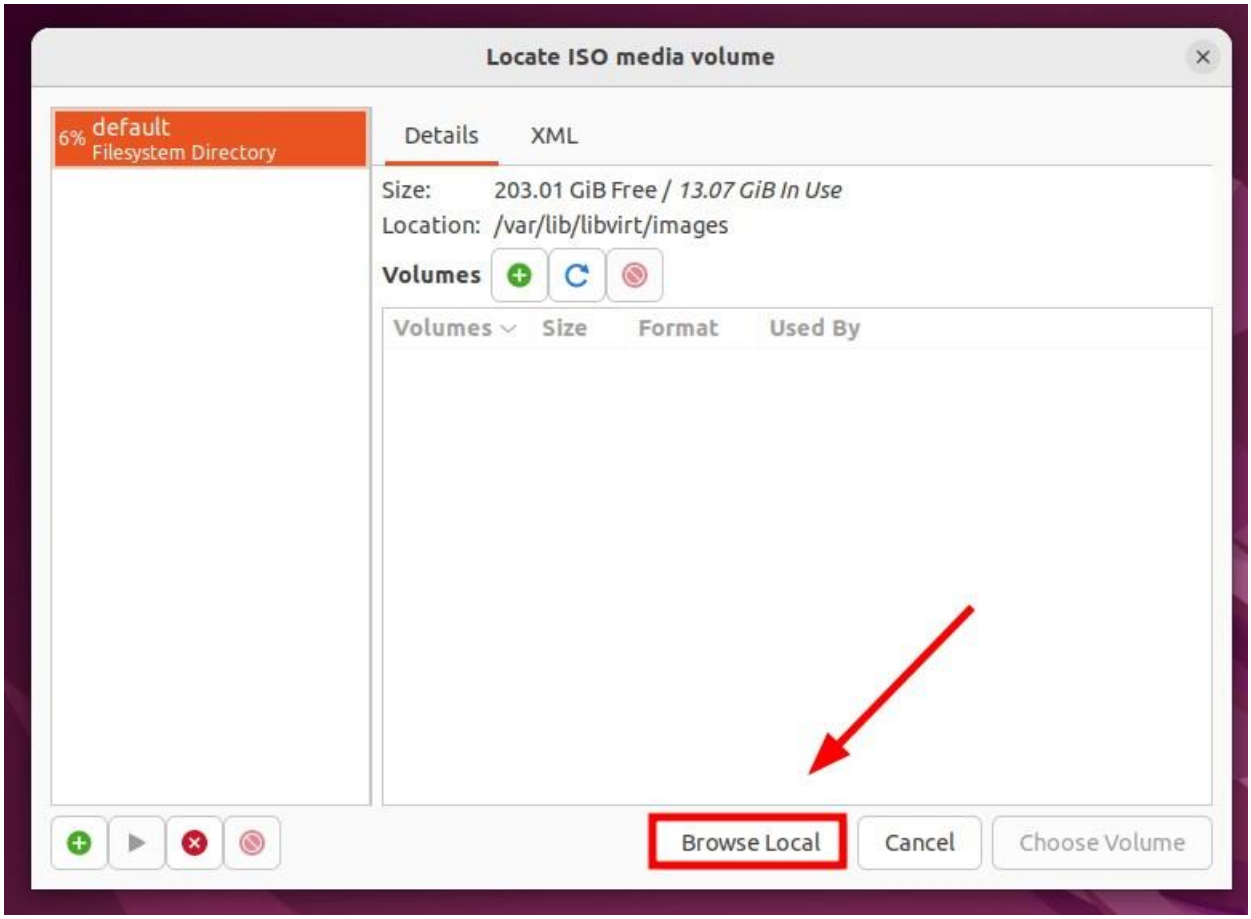
Choose VM Install Type

Next click '**Browse**' to navigate to the location of the ISO file.



Choose VM ISO File

Since the ISO file is saved locally on your system, we will click '**Browse Local**'.



Browse Local Filesystem

Be sure to navigate to the location of your ISO file. Click it and then click '**Open**'.

Choose OS ISO File.

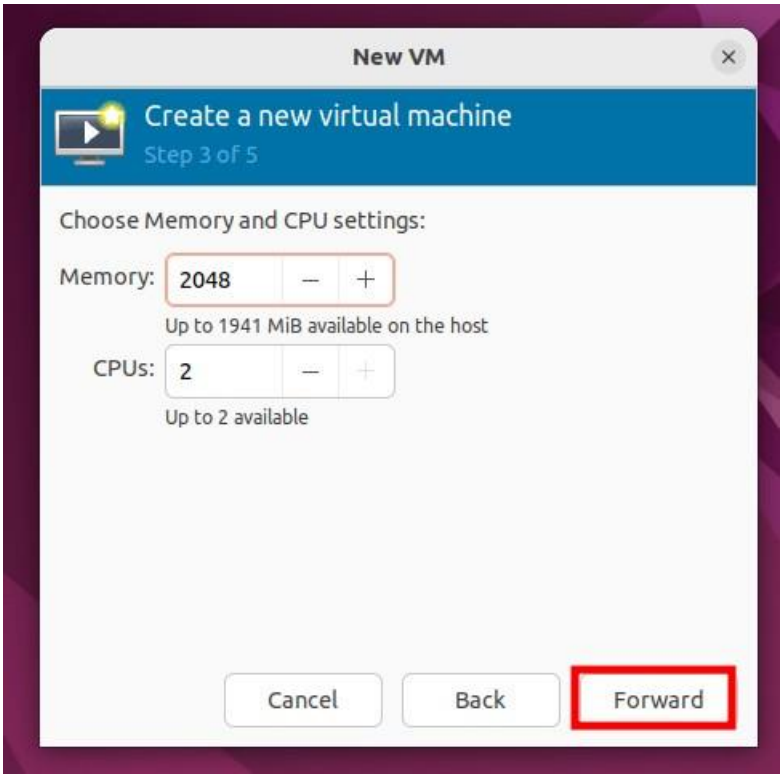
Before proceeding, ensure that you have selected the operating system from the drop-down menu. Then click '**Forward**'.

Choose VM Operating System.

Click '**Yes**' on the pop-up to grant the emulator search permissions to the ISO file.

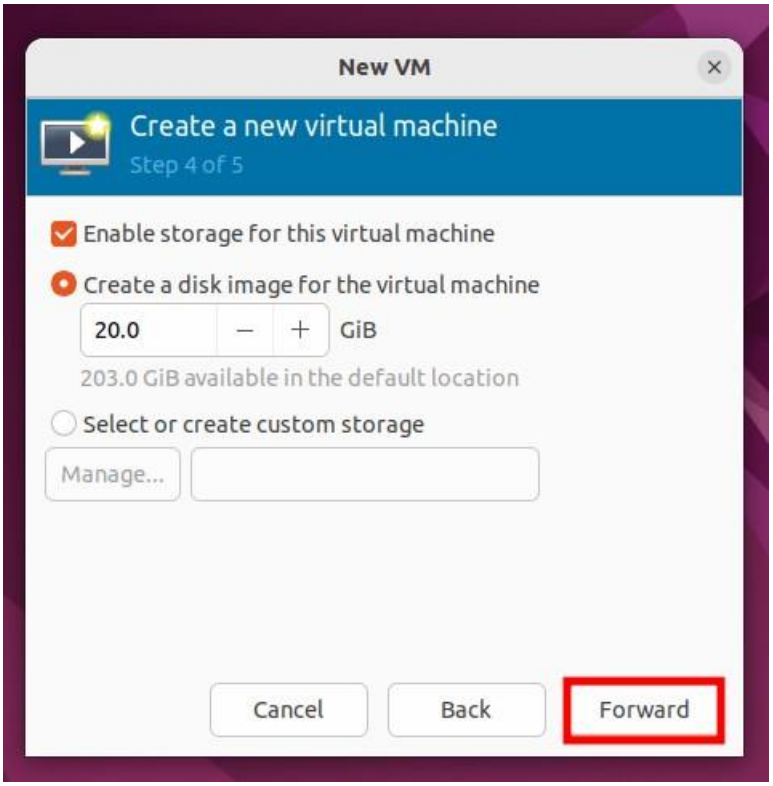
Grant Permission on Emulator

Next, select the Memory size and the number of CPU cores and click '**Forward**'.



Choose VM Memory and CPU Settings

In the next step, enable storage for the virtual machine and specify the virtual disk size. Then click '**Forward**'.

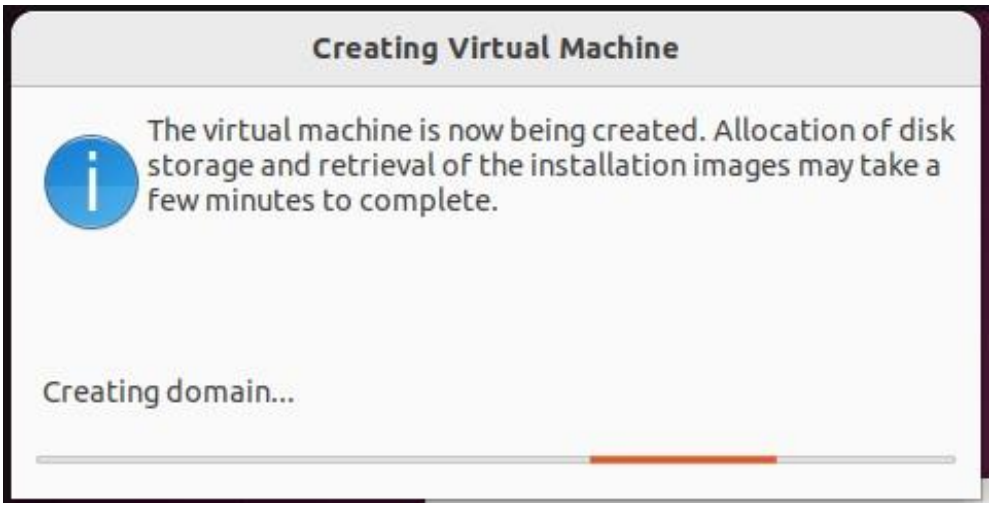


Choose VM Disk Size

Finally, review all the settings that you have defined, and if all looks good click '**Finish**' to create the virtual machine. Else click '**back**' and make the necessary changes.

Review VM Settings

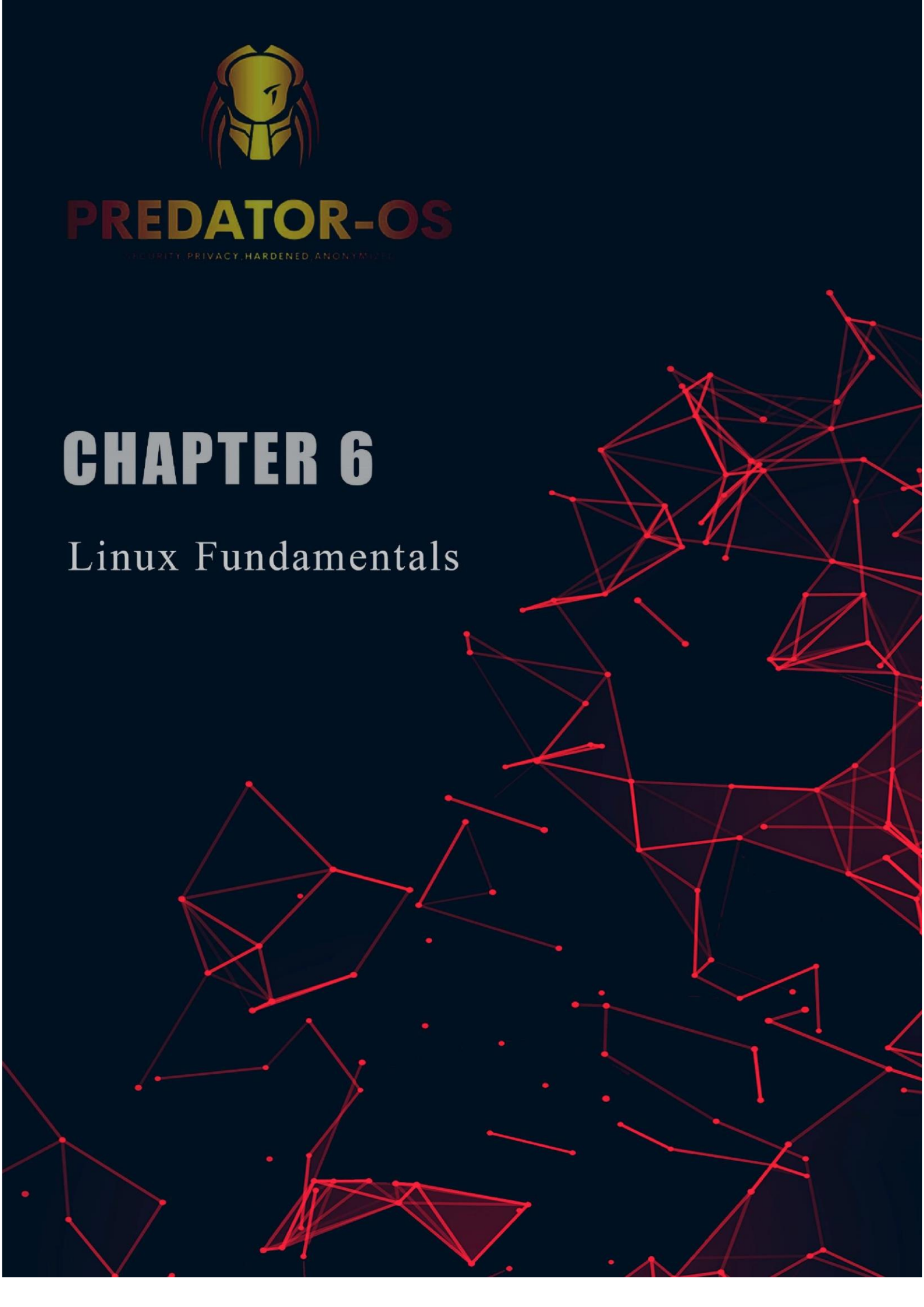
Once you click '**Finish**' the virtual machine manager will start creating the virtual machine based on the set configurations.



Creating a Virtual Machine in Qemu.

And in a matter of seconds, the virtual machine installation wizard will pop up. You can proceed with the installation as you would on a physical system.

Installing Predator-OS in VMWare will be beginning now.



CHAPTER 6

Linux Fundamentals

Chapter 6

Linux Fundamental

This chapter takes look at the most common directories in the **Linux file tree**. It also shows that on Unix everything is a file.

filesystem hierarchy standard

Many Linux distributions partially follow the **Filesystem Hierarchy Standard**. The **FHS** may help make more Unix/Linux file system trees conform better in the future. The **FHS** is available online at **http://www.pathname.com/fhs/** where we read: "The filesystem hierarchy standard has been designed to be used by Unix distribution developers, package developers, and system implementers. However, it is primarily intended to be a reference and is not a tutorial on how to manage a Unix filesystem or directory hierarchy."

The root directory /

All Linux systems have a directory structure that starts at the **root directory**. The root directory is represented by a **forward slash**, like this: /. Everything that exists on your Linux system can be found below this root directory. Let's take a brief look at the contents of the root directory.

```
[user@Predator-pc]$ ls  /
bin  dev  home  media  mnt  proc  sbin   srv  tftpboot  usr boot  etc  lib  misc  opt  root  selinux  sys  tmp  var
```

Binary directories

Binaries are files that contain compiled source code (or machine code). Binaries can be **executed** on the computer. Sometimes binaries are called **executables**.

/bin

The **/bin** directory contains **binaries** for use by all users. According to the FHS the **/bin** directory should contain **/bin/cat** and **/bin/date** (among others). you see common Unix/Linux commands like cat, cp, cpio, date, dd, echo, grep, and so on. Many of these will be covered in this book.

```
user@predator-pc~$ ls /bin
archdetect egrep mt setupcon autopartition false mt-gnu sh bash
fgconsole mv sh.distrib bunzip2 fgrep nano sleep bzcat fuser nc
stralign bzcmp fusermount nc.traditional stty bzdiff get_mountoptions netcat su bzegrep
grep netstat sync bzexe gunzip ntfs-3g sysfs bzfgrep gzexe ntfs-
3g.probe tailf bzgrep gzip parted_devices tar bzip2 hostname parted_server
tempfile bzip2recover hw-detect partman touch bzless ip partman-commit true
bzmore kbd_mode perform_recipe ulockmgr cat kill pidof umount ...
```

Other /bin directories

You can find a **/bin subdirectory** in many other directories. A user named **serena** could put her own programs in **/home/serena/bin**.

Some applications, often when installed directly from source will put themselves in **/opt**. A **samba server** installation can use **/opt/samba/bin** to store its binaries.

/sbin

/sbin contains binaries to configure the operating system. Many of the **system binaries** require **root** privilege to perform certain tasks.

Below a screenshot containing **system binaries** to change the ip address, partition a disk and create an ext4 file system.

```
user@ predator-pc:~$ ls -l /sbin/ifconfig /sbin/fdisk /sbin/mkfs.ext4
-rwxr-xr-x 1 root root 97172 2011-02-02 09:56 /sbin/fdisk
-rwxr-xr-x 1 root root 65708 2010-07-02 09:27 /sbin/ifconfig -rwxr-xr-x 5 root root 55140 2010-
08-18 18:01 /sbin/mkfs.ext4
```

/lib

Binaries found in **/bin** and **/sbin** often use **shared libraries** located in **/lib**. Below is a screenshot of the partial contents of **/lib**.

```
user@predator-pc~$ ls /lib/libc*
/lib/libc-2.5.so      /lib/libcfont.so.0.0.0  /lib/libcom_err.so.2.1
/lib/libcap.so.1      /lib/libcidn-2.5.so     /lib/libconsole.so.0
/lib/libcap.so.1.10   /lib/libcidn.so.1       /lib/libconsole.so.0.0.0
/lib/libcfont.so.0    /lib/libcom_err.so.2    /lib/libcrypt-2.5.so
```

/lib/modules

Typically, the **Linux kernel** loads kernel modules from **/lib/modules/$kernelversion/**. This directory is discussed in detail in the Linux kernel chapter.

/lib32 and /lib64

We currently are in a transition between **32-bit** and **64-bit** systems. Therefore, you may encounter directories named **/lib32** and **/lib64** which clarify the register size used during compilation time of the libraries. A 64-bit computer may have some 32-bit binaries and libraries for compatibility with legacy applications. This screenshot uses the **file** utility to demonstrate the difference.

```
user@predator-pc~$ file /lib32/libc-2.5.so


/lib32/libc-2.5.so: ELF 32-bit LSB shared object, Intel 80386, \ version 1
(SYSV), for GNU/Linux 2.6.0, stripped user@predator-pc~$ file
/lib64/libcap.so.1.10
/lib64/libcap.so.1.10: ELF 64-bit LSB shared object, AMD x86-64, \ version
1 (SYSV), stripped
```

The ELF (**Executable and Linkable Format**) is used in almost every Unix-like operating system since **System V**.

/opt

The purpose of **/opt** is to store **optional** software. In many cases this is software from outside the distribution repository. You may find an empty **/opt** directory on many systems.

A large package can install all its files in **/bin**, **/lib**, **/etc** subdirectories within **/opt/$packagename/**. If for example the package is called wp, then it installs in **/opt/wp**, putting binaries in **/opt/wp/bin** and manpages in **/opt/wp/man**.

configuration directories

/boot

The **/boot** directory contains all files needed to boot the computer. These files don't change very often. On Linux systems you typically find the **/boot/grub** directory here. **/boot/grub** contains **/boot/grub/grub.cfg** (older systems may still have **/boot/grub/grub.conf**) which defines the boot menu that is displayed before the kernel starts.

/etc

All of the machine-specific **configuration files** should be located in **/etc**. Historically **/etc** stood for **etcetera**, today people often use the **Editable Text Configuration** backronym.

Many times the name of a configuration files is the same as the application, daemon, or protocol with **.conf** added as the extension.

```
user@predator-pc~$ ls /etc/*.conf
/etc/adduser.conf        /etc/ld.so.conf       /etc/scrollkeeper.conf
/etc/brltty.conf         /etc/lftp.conf        /etc/sysctl.conf
/etc/ccertificates.conf  /etc/libao.conf       /etc/syslog.conf
```

```
/etc/cvs-cron.conf      /etc/logrotate.conf   /etc/ucf.conf
/etc/ddclient.conf      /etc/ltrace.conf      /etc/uniconf.conf
/etc/debconf.conf       /etc/mke2fs.conf      /etc/updatedb.conf
/etc/deluser.conf       /etc/netscsid.conf    /etc/usplash.conf
/etc/fdmount.conf       /etc/nsswitch.conf    /etc/uswsusp.conf
/etc/hdparm.conf        /etc/pam.conf         /etc/vnc.conf
/etc/host.conf          /etc/pnm2ppa.conf     /etc/wodim.conf
/etc/inetd.conf         /etc/povray.conf      /etc/wvdial.conf
/etc/kernel-img.conf    /etc/resolv.conf user@predator-pc~$
```

There is much more to be found in **/etc**.

/etc/init.d/

A lot of Unix/Linux distributions have an **/etc/init.d** directory that contains scripts to start and stop **daemons**. This directory could disappear as Linux migrates to systems that replace the old **init** way of starting all **daemons**.

/etc/X11/

The graphical display (aka **X Window System** or just **X**) is driven by software from the X.org foundation. The configuration file for your graphical display is **/etc/X11/xorg.conf**. **/etc/skel/**

The **skeleton** directory **/etc/skel** is copied to the home directory of a newly created user. It usually contains hidden files like a **.bashrc** script.

/etc/sysconfig/

This directory; which is not mentioned in the FHS, contains a lot of Red Hat Enterprise Linux configuration files. We will discuss some of them in greater detail.

```
user@Predator-pc:~$ ls /etc/sysconfig/
apmd            firstboot       irda                  network        saslauthd
apmscripts  grub          irqbalance        networking   selinux authconfig
hidd          keyboard          ntpd         spamassassin autofs       httpd
kudzu              openib.conf   squid bluetooth     hwconf        lm_sensors
pand          syslog clock         i18n           mouse              pcmcia
sys-config-sec console         init              mouse.B              pgsql
sysconfig-users crond          installinfo    named                prelink
syslogviewer
desktop       ipmi           netdump           rawdevices    tux
diskdump      iptables       netdump_id_dsa    rhn           vncservers
dund          iptables-cfg netdump_id_dsa.p  samba         xinetd
user@Predator-pc:~$
```

The file **/etc/sysconfig/firstboot** tells the Red Hat Setup Agent not to run at boot time. If you want to run the Red Hat Setup Agent at the next reboot, then simply remove this file, and run **chkconfig --level 5 firstboot on**. The Red Hat Setup Agent allows you to install the latest updates, create a user account, join the Red Hat Network and more. It will then create the /etc/sysconfig/firstboot file again.

user@Predator-pc:~$ cat /etc/sysconfig/firstboot RUN_FIRSTBOOT=NO

The **/etc/sysconfig/harddisks** file contains some parameters to tune the hard disks. The file explains itself.

You can see hardware detected by **kudzu** in **/etc/sysconfig/hwconf**. Kudzu is software from Red Hat for automatic discovery and configuration of hardware.

The keyboard type and keymap table are set in the **/etc/sysconfig/keyboard** file. For more console keyboard information, check the manual pages of **keymaps(5)**, **dumpkeys(1)**, **loadkeys(1)** and the directory **/lib/kbd/keymaps/**.

```
root@Predator-pc:/etc/sysconfig# cat keyboard
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

We will discuss networking files in this directory in the networking chapter. **data directories**

/home

Users can store personal or project data under **/home**. It is common (but not mandatory by the fhs) practice to name the users home directory after the user name in the format **/home/ $USERNAME**. For example:

Besides giving every user (or every project or group) a location to store personal files, the home directory of a user also serves as a location to store the user profile. A typical Unix user profile contains many hidden files (files whose file name starts with a dot). The hidden files of the Unix user profiles contain settings specific for that user.

```
user@Predator-pc::~$ ls -d /home/user/.*
/home/user/.              /home/user/.bash_profile  /home/user/.ssh
/home/user/..             /home/user/.bashrc        /home/user/.viminfo
/home/user/.bash_history  /home/user/.lesshst
```

/root

On many systems **/root** is the default location for personal data and profile of the **root user**. If it does not exist by default, then some administrators create it.

/srv

You may use **/srv** for data that is **served by your system**. The FHS allows locating cvs, rsync, ftp and www data in this location. The FHS also approves administrative naming in / srv, like /srv/project55/ftp and /srv/sales/www.

On Sun Solaris (or Oracle Solaris) **/export** is used for this purpose.

/media

The /**media** directory serves as a mount point for **removable media devices** such as CDROM's, digital cameras, and various usb-attached devices. Since /**media** is rather new in the Unix world, you could very well encounter systems running without this directory. Solaris 9 does not have it, Solaris 10 does. Most Linux distributions today mount all removable media in /**media**.

```
user@predator:~$ ls /media/ cdrom  cdrom0  usbdisk
```

/mnt

The /**mnt** directory should be empty and should only be used for temporary mount points (according to the FHS).

Unix and Linux administrators used to create many directories here, like /mnt/something/. You likely will encounter many systems with more than one directory created and/or mounted inside /**mnt** to be used for various local and remote filesystems.

/tmp

Applications and users should use /**tmp** to store temporary data when needed. Data stored in /**tmp** may use either disk space or RAM. Both of which are managed by the operating system. Never use /**tmp** to store data that is important or which you wish to archive.

in memory directories

/dev

Device files in /**dev** appear to be ordinary files but are not actually located on the hard disk. The /**dev** directory is populated with files as the kernel is recognising hardware.

common physical devices

Common hardware such as hard disk devices are represented by device files in /**dev**. Below a screenshot of SATA device files on a laptop and then IDE attached drives on a desktop. (The detailed meaning of these devices will be discussed later.)

```
#
# SATA or SCSI or USB
#
user@predator-pc~$ ls /dev/sd* /dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sdb
/dev/sdb1  /dev/sdb2

#
# IDE or ATAPI
```

```
#
user@barry:~$ ls /dev/hd*
/dev/hda  /dev/hda1  /dev/hda2  /dev/hdb  /dev/hdb1  /dev/hdb2  /dev/hdc
```

Besides representing physical hardware, some device files are special. These special devices can be very useful.

/dev/tty and /dev/pts

For example, /**dev**/**tty1** represents a terminal or console attached to the system. (Don't break your head on the exact terminology of 'terminal' or 'console', what we mean here is a command line interface.) When typing commands in a terminal that is part of a graphical interface like Gnome or KDE, then your terminal will be represented as /**dev**/**pts**/**1** (1 can be another number).

/dev/null

On Linux you will find other special devices such as /**dev**/**null** which can be considered a black hole; it has unlimited storage, but nothing can be retrieved from it. Technically speaking, anything written to /dev/null will be discarded. /dev/null can be useful to discard unwanted output from commands. */dev/null is not a good location to store your backups ;-).*

/proc conversation with the kernel

/**proc** is another special directory, appearing to be ordinary files, but not taking up disk space. It is actually a view of the kernel, or better, what the kernel manages, and is a means to interact with it directly. /**proc** is a proc filesystem.

```
user@Predator-pc:~$ mount -t proc
```

```
none on /proc  type proc (rw)
```

When listing the /proc directory you will see many numbers (on any Unix) and some interesting files (on Linux)

```
user@predator-pc~$ ls /proc
1      2339   4724  5418  6587  7201       cmdline      mounts
10175  2523   4729  5421  6596  7204       cpuinfo      mtrr
10211  2783   4741  5658  6599  7206       crypto       net
10239  2975   4873  5661  6638  7214       devices      pagetypeinfo
141    29775  4874  5665  6652  7216       diskstats    partitions
15045  29792  4878  5927  6719  7218       dma          sched_debug
1519   2997   4879  6     6736  7223       driver       scsi
1548   3      4881  6032  6737  7224       execdomains  self
1551   30228  4882  6033  6755  7227       fb           slabinfo
1554   3069   5     6145  6762  7260       filesystems  stat
1557   31422  5073  6298  6774  7267       fs           swaps
1606   3149   5147  6414  6816  7275       ide          sys
180    31507  5203  6418  6991  7282       interrupts   sysrq-trigger
181    3189   5206  6419  6993  7298       iomem        sysvipc
182    3193   5228  6420  6996  7319       ioports      timer_list
18898  3246   5272  6421  7157  7330       irq          timer_stats 19799  3248
5291   6422   7163  7345       kallsyms     tty
```

```
19803  3253   5294  6423  7164  7513       kcore        uptime
19804  3372   5356  6424  7171  7525       key-users    version
1987   4      5370  6425  7175  7529       kmsg         version_signature
1989   42     5379  6426  7188  9964       loadavg      vmcore
2      45     5380  6430  7189  acpi       locks        vmnet
20845  4542   5412  6450  7191  asound     meminfo      vmstat
221    46     5414  6551  7192  buddyinfo  misc         zoneinfo
2338   4704   5416  6568  7199  bus        modules
```

Let's investigate the file properties inside /**proc**. Looking at the date and time will display the current date and time showing the files are constantly updated (a view on the kernel).

```
user@Predator-pc:~$ date Mon Jan 29
18:06:32 EST 2007 user@Predator-pc:~$ ls
-al /proc/cpuinfo
-r--r--r--  1 root root 0 Jan 29 18:06 /proc/cpuinfo
user@Predator-pc:~$ user@Predatorpc:~$
      ...time      passes...
user@Predator-pc:~$   user@Predatorpc:~$
date
Mon Jan 29 18:10:00 EST 2007 user@Predator-pc:~$
ls -al /proc/cpuinfo
-r--r--r--  1 root root 0 Jan 29 18:10 /proc/cpuinfo
```

Most files in /proc are 0 bytes, yet they contain data--sometimes a lot of data. You can see this by executing cat on files like /**proc**/**cpuinfo**, which contains information about the CPU.

```
user@Predator-pc:~$ file /proc/cpuinfo
/proc/cpuinfo: empty
user@Predator-pc:~$ cat /proc/cpuinfo  processor       : 0 vendor_id       :
AuthenticAMD cpu family      : 15 model

: 43
```

Just for fun, here is /proc/cpuinfo on a Sun Sunblade 1000...

```
user@pasha:~$ cat /proc/cpuinfo cpu : TI UltraSparc III (Cheetah) fpu :
UltraSparc III integrated FPU promlib : Version 3 Revision 2 prom : 4.2.2 type :
sun4u ncpus probed : 2 ncpus active  :      2

Cpu0Bogo : 498.68
Cpu0ClkTck : 000000002cb41780 Cpu1Bogo : 498.68
Cpu1ClkTck : 000000002cb41780
MMU Type : Cheetah State:
CPU0: online
CPU1: online
```

Most of the files in /proc are read only, some require root privileges, some files are writable, and many files in /**proc**/**sys** are writable. Let's discuss some of the files in /proc.

/proc/interrupts

On the x86 architecture, /**proc**/**interrupts** displays the interrupts.

```
user@Predator-pc:~$ cat /proc/interrupts
           CPU0
  0:   13876877    IO-APIC-edge  timer
  1:         15    IO-APIC-edge  i8042
  8:          1    IO-APIC-edge  rtc
  9:          0   IO-APIC-level  acpi
 12:         67    IO-APIC-edge  i8042
 14:        128    IO-APIC-edge  ide0
 15:     124320    IO-APIC-edge  ide1
169:     111993   IO-APIC-level  ioc0
177:       2428   IO-APIC-level  eth0
NMI:          0
LOC:   13878037
ERR:          0
MIS:          0
```

On a machine with two CPU's, the file looks like this.

```
user@predator-pc~$ cat /proc/interrupts
           CPU0       CPU1
  0:     860013          0  IO-APIC-edge     timer
  1:       4533          0  IO-APIC-edge     i8042
  7:          0          0  IO-APIC-edge     parport0
  8:    6588227          0  IO-APIC-edge     rtc
 10:       2314          0  IO-APIC-fasteoi  acpi
 12:        133          0  IO-APIC-edge     i8042
 14:          0          0  IO-APIC-edge     libata
 15:      72269          0  IO-APIC-edge     libata
 18:          1          0  IO-APIC-fasteoi  yenta
 19:     115036          0  IO-APIC-fasteoi  eth0
 20:     126871          0  IO-APIC-fasteoi  libata, ohci1394
 21:      30204          0  IO-APIC-fasteoi  ehci_hcd:usb1, uhci_hcd:usb2
 22:       1334          0  IO-APIC-fasteoi  saa7133[0], saa7133[0]
 24:     234739          0  IO-APIC-fasteoi  nvidia
NMI:         72         42
LOC:     860000     859994
ERR:          0
```

/proc/kcore

The physical memory is represented in **/proc/kcore**. Do not try to cat this file instead use a debugger. The size of /proc/kcore is the same as your physical memory plus four bytes.

```
user@predator-pc~$ ls -lh /proc/kcore
-r-------- 1 root root 2.0G 2007-01-30 08:57 /proc/kcore user@predator-pc~$
```

/sys Linux hot plugging

The **/sys** directory was created for the Linux 2.6 kernel. Since 2.6, Linux uses **sysfs** to support **usb** and **IEEE 1394** (**FireWire**) hot plug devices. See the manual pages of udev(8) (the successor of **devfs**) and hotplug(8) for more info (or visit http://linuxhotplug.sourceforge.net/).

Basically the **/sys** directory contains kernel information about hardware.

/usr Unix System Resources

Although /**usr** is pronounced like user remember that it stands for **Unix System Resources**. The /**usr** hierarchy should contain **shareable read only** data. Some people choose to mount /**usr** as read only. This can be done from its own partition or from a read only NFS share (NFS is discussed later).

/usr/bin

The /**usr**/**bin** directory contains a lot of commands.

```
user@deb508:~$          ls

/usr/bin | wc -l 1395
```

(On Solaris the /**bin** directory is a symbolic link to /**usr**/**bin**.)

/usr/include

The /**usr**/**include** directory contains general use include files for C.

```
user@ubu1010:~$ ls /usr/include/
aalib.h           expat_config.h        math.h              searc
expat_external.h     mcheck.h           semaphore.h aio.h
memory.h             setjmp. h AL             fcntl.h
sgtty.h aliases. h       features.h          mntent.h          sh
```

/usr/lib

The /**usr**/**lib** directory contains libraries that are not directly executed by users or scripts.

```
user@deb508:~$ ls /usr/lib | head -7
4Suite
ao apt
arj
aspell
avahi
bonobo
```

/usr/local

The /**usr**/**local** directory can be used by an administrator to install software locally.

```
user@deb508:~$ ls /usr/local/
bin  etc  games  include  lib  man  sbin  share  src user@deb508:~$ du -sh
/usr/local/ 128K /usr/local/
```

/usr/share

The /**usr**/**share** directory contains architecture independent data. As you can see, this is a fairly large directory.

```
user@deb508:~$ ls /usr/share/ | wc -l
```

```
263 user@deb508:~$ du -sh /usr/share/ 1.3G
/usr/share/
```

This directory typically contains **/usr/share/man** for manual pages.

```
user@deb508:~$ ls /usr/share/man
cs  fr    hu  it.UTF-8  man2 man6  pl.ISO8859-2  sv de  fr.ISO8859-1  id  ja   man3
man7  pl.UTF-8      tr es   fr.UTF8   it   ko    man4 man8  pt_BR      zh_CN fi  gl
it.ISO8859-

1 man1   man5 pl    ru      zh_TW
```

And it contains **/usr/share/games** for all static game data (so no high-scores or play logs).

```
user@ubu1010:~$ ls
/usr/share/games/
```

/usr/src

The **/usr/src** directory is the recommended location for kernel source files.

```
user@deb508:~$ ls -l /usr/src/
```

/var variable data

Files that are unpredictable in size such as log, cache and spool files, should be located in **/var**.

/var/log

The **/var/log** directory serves as a central point to contain all log files.

set

You can use the **set** command to display a list of environment variables. On Debian stable and Debian systems, the set command will also list shell functions after the shell variables. Use set | more to see the variables then.

unset

Use the **unset** command to remove a variable from your shell environment.

```
[user@RHEL4b ~]$ MyVar=8472
[user@RHEL4b ~]$ echo $MyVar
8472
[user@RHEL4b ~]$ unset MyVar
[user@RHEL4b ~]$ echo $MyVar

[user@RHEL4b ~]$
```

PS1

The **$PS1** variable determines your shell prompt. You can use backslash escaped special characters like **\u** for the username or **\w** for the working directory. The **bash** manual has a complete reference.

In this example we change the value of **$PS1** a couple of times.

```
user@deb503:~$ PS1=prompt prompt promptPS1='prompt ' prompt prompt
PS1='> ' >
> PS1='\u@\h$ ' user@deb503$
user@deb503$ PS1='\u@\h:\W$' user@deb503:~$
```

To avoid unrecoverable mistakes, you can set normal user prompts to green and the root prompt to red. Add the following to your **.bashrc** for a green user prompt:

```
# color prompt by user
RED='\[\033[01;31m\]'
WHITE='\[\033[01;00m\]'
GREEN='\[\033[01;32m\]'
BLUE='\[\033[01;34m\]' export

PS1="${debian_chroot:+($debian_chroot)}$GREEN\u$WHITE@$BLUE\h$WHITE\w\$ "
```

$PATH

The **$PATH** variable is determines where the shell is looking for commands to execute (unless the command is builtin or aliased). This variable contains a list of directories, separated by colons.

```
[[user@RHEL4b ~]$ echo $PATH /usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
```

The shell will not look in the current directory for commands to execute! (Looking for executables in the current directory provided an easy way to hack PC-DOS computers). If you want the shell to look in the current directory then add a **. (dot)** at the end of your $PATH.

```
[user@RHEL4b ~]$ PATH=$PATH:.
[user@RHEL4b ~]$ echo $PATH /usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:.
[user@RHEL4b ~]$
```

Your path might be different when using su instead of **su** - because the latter will take on the environment of the target user. The root user typically has **/sbin** directories added to the $PATH variable.

```
[user@RHEL3 ~]$   su

Password:
[root@RHEL3 user]# echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
[root@RHEL3 user]# exit
[user@RHEL3 ~]$ su Password:
[root@RHEL3 ~]# echo $PATH

/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin: [root@RHEL3 ~]#
```

env

The **env** command without options will display a list of **exported variables**. The difference with **set** with options is that **set** lists all variables, including those not exported to child shells.

But **env** can also be used to start a clean shell (a shell without any inherited environment). The **env -i** command clears the environment for the subshell. Notice in this screenshot that **bash** will set the **$SHELL** variable on startup.

```
[user@RHEL4b ~]$ bash -c 'echo $SHELL $HOME $USER'
/bin/bash /home/user user
[user@RHEL4b ~]$ env -i bash -c 'echo $SHELL $HOME $USER'
/bin/bash
[user@RHEL4b ~]$
```

You can use the **env** command to set the **$LANG**, or any other, variable for just one instance of **bash** with one command. The example below uses this to show the influence of the **$LANG** variable on file globbing (see the chapter on file globbing).

```
[user@RHEL4b test]$ env LANG=C bash -c 'ls File[a-z]'
Filea  Fileb
[user@RHEL4b test]$ env LANG=en_US.UTF-8 bash -c 'ls File[a-z]'
Filea  FileA  Fileb  FileB [user@RHEL4b test]$
```

export

You can export shell variables to other shells with the **export** command. This will export the variable to child shells.

```
[user@RHEL4b ~]$ var3=three
[user@RHEL4b ~]$ var4=four
[user@RHEL4b ~]$ export var4 [user@RHEL4b ~]$ echo $var3 $var4 three four
[user@RHEL4b ~]$ bash
[user@RHEL4b ~]$ echo $var3 $var4 four
```

But it will not export to the parent shell (previous screenshot continued).

```
[user@RHEL4b ~]$ export var5=five
[user@RHEL4b ~]$ echo $var3 $var4 $var5 four
five
[user@RHEL4b ~]$ exit exit
[user@RHEL4b ~]$ echo $var3 $var4
$var5 three four [user@RHEL4b ~]$
```

Licensing

About software licenses

There are two predominant software paradigms: **Free and Open Source Software** (FOSS) and **proprietary software**. The criteria for differentiation between these two approaches is based on control over the software. With **proprietary software** control tends to lie more with the vendor, while with **Free and Open Source Software** tends to be more weighted towards the end user. But even though the paradigms differ, they use the same **copyright laws** to reach and enforce their goals. From a legal perspective, **Free and Open Source Software** can be considered as software to which users generally receive more rights via their license agreement than they would have with a **proprietary software license**, yet the underlying license mechanisms are the same.

Legal theory states that the author of FOSS, contrary to the author of **public domain** software, has in no way whatsoever given up his rights on his work. FOSS supports on the rights of the author (the **copyright**) to impose FOSS license conditions. The FOSS license conditions need to be respected by the user in the same way as proprietary license conditions. Always check your license carefully before you use third party software.

Examples of proprietary software are **AIX** from IBM, **HP-UX** from HP and **Oracle Database 11g**. You are not authorised to install or use this software without paying a licensing fee. You are not authorised to distribute copies and you are not authorised to modify the closed source code.

Public domain software and freeware

Software that is original in the sense that it is an intellectual creation of the author benefits **copyright** protection. Non-original software does not come into consideration for **copyright** protection and can, in principle, be used freely. Public domain software is considered as software to which the author has given up all rights and on which nobody is able to enforce any rights. This software can be used, reproduced or executed freely, without permission or the payment of a fee. Public domain software can in certain cases even be presented by third parties as own work, and by modifying the original work, third parties can take certain versions of the public domain software out of the public domain again.

Freeware is not public domain software or FOSS. It is proprietary software that you can use without paying a license cost. However, the often-strict license terms need to be respected.

Examples of freeware are **Adobe Reader**, **Skype** and **Command and Conquer**: **Tiberian Sun** (this game was sold as proprietary in 1999 and is since 2011 available as freeware).

Free Software or Open Source Software

Both the **Free Software** (translates to **vrije software** in Dutch and to **Logiciel Libre** in French) and the **Open-Source Software** movement largely pursue similar goals and endorse similar software licenses. But historically, there has been some perception of differentiation due to different emphases. Where the **Free Software** movement focuses on the rights (the four freedoms) which Free Software provides to its users, the **Open Source Software** movement points to its Open Source Definition and the advantages of peer-to-peer software development. Recently, the term free and open source software or FOSS has arisen as a neutral alternative. A lesser-used variant is free/libre/open source software (FLOSS), which uses **libre** to clarify the meaning of free as in **freedom** rather than as in **at no charge**.

Examples of **free software** are **gcc**, **MySQL** and **gimp**.

Detailed information about the **four freedoms** can be found here:

http://www.gnu.org/philosophy/free-sw.html

The **open-source definition** can be found at:

http://www.opensource.org/docs/osd

The above definition is based on the **Debian Free Software Guidelines** available here:

http://www.debian.org/social_contract#guidelines

GNU General Public License

More and more software is being released under the **GNU GPL** (in 2006 Java was released under the GPL). This license (v2 and v3) is the main license endorsed by the Free Software Foundation. It's main characteristic is the **copyleft** principle. This means that everyone in the chain of consecutive users; in return for the right of use that is assigned, needs to distribute the improvements he makes to the software and his derivative works under the same conditions to other users, if he chooses to distribute such improvements or derivative works. In other words; software which incorporates GNU GPL software, needs to be distributed in turn as GNU GPL software (or compatible, see below). It is not possible to incorporate copyright protected parts of GNU GPL software in a proprietary licensed work. The GPL has been upheld in court.

Using GPLv3 software

You can use **GPLv3 software** almost without any conditions. If you solely run the software, you even don't have to accept the terms of the GPLv3. However, any other use - such as modifying or distributing the software - implies acceptance. In case you use the software internally (including over a network), you may modify the software without being obliged to distribute your modification. You may hire third parties to work on the software exclusively for you and under your direction and control. But if you modify the software and use it otherwise than merely internally, this will be considered as distribution. You must distribute your modifications under GPLv3 (the copyleft principle). Several more obligations apply if you distribute GPLv3 software. Check the GPLv3 license carefully. You create output with GPLv3 software: The GPLv3 does not automatically apply to the output.

BSD license

There are several versions of the original Berkeley Distribution License. The most common one is the 3-clause license ("New BSD License" or "Modified BSD License").

This is a permissive free software license. The license places minimal restrictions on how the software can be redistributed. This is in contrast to copyleft licenses such as the GPLv. 3 discussed above, which have a copyleft mechanism. This difference is of less importance when you merely use the software, but kicks in when you start redistributing verbatim copies of the software or your own modified versions.

Other licenses

FOSS or not, there are many kind of licenses on software. You should read and understand them before using any software.

Combination of software licenses

When you use several sources or wishes to redistribute your software under a different license, you need to verify whether all licenses are compatible. Some FOSS licenses (such as BSD) are compatible with proprietary licenses, but most are not. If you detect a license incompatibility, you must contact the author to negotiate different license conditions or refrain from using the incompatible software.

First steps on the command line man pages

This chapter will explain the use of **man** pages (also called **manual pages**) on your Unix or Linux computer.

You will learn the **man** command together with related commands like **whereis**, **whatis** and **mandb**.
Most Unix files and commands have pretty good man pages to explain their use. Man pages also come in handy when you are using multiple flavours of Unix or several Linux distributions since options and parameters sometimes vary.

man $command

Type **man** followed by a command (for which you want help) and start reading. Press **q** to quit the manpage. Some man pages contain examples (near the end).

```
user@predator-pc~$      man    whois Reformatting
whois(1), please wait...
```

man $configfile

Most **configuration files** have their own manual.

```
user@predator-pc~$      man    syslog.conf Reformatting
syslog.conf(5), please wait...
```

man $daemon

This is also true for most **daemons** (background programs) on your system.

```
user@predator-pc~$      man    syslogd Reformatting
syslogd(8), please wait...
```

man -k (apropos) man -k (or apropos) shows a list of man pages containing a string.

```
user@predator-pc~$ man -k syslog lm-syslog-setup (8)  - configure laptop
mode to switch syslog.conf ... logger (1)             - a shell command
interface to the syslog(3) ... syslog-facility (8)   - Setup and remove
LOCALx facility for sysklogd syslog.conf (5)          - syslogd(8)
configuration file syslogd (8)        - Linux system logging utilities.
syslogdlistfiles (8) - list system logfiles
```

whatis

To see just the description of a manual page, use **whatis** followed by a string.

```
user@u810:~$ whatis route
route (8)             - show / manipulate the IP routing table
```

whereis

The location of a manpage can be revealed with **whereis**.

```
user@predator-pc~$ whereis -m whois whois:
/usr/share/man/man1/whois.1.gz
```

This file is directly readable by **man**.

```
user@predator-pc~$ man /usr/share/man/man1/whois.1.gz
```

man $section $file

Therefor, when referring to the man page of the passwd command, you will see it written as **passwd(1);** when referring to the **passwd file**, you will see it written as **passwd(5)**. The screenshot explains how to open the man page in the correct section.

```
[user@RHEL52 ~]$ man passwd      # opens the first manual found
[user@RHEL52 ~]$ man 5 passwd    # opens a page from section 5
```

man man

If you want to know more about **man**, then Read The Fantastic Manual (RTFM).

Unfortunately, manual pages do not have the answer to everything...

```
user@predator-pc~$      man
woman No manual entry for woman
```

mandb

Should you be convinced that a man page exists, but you can't access it, then try running **mandb** on Debian/Mint.

```
root@predator-pc~# mandb 0 man subdirectories contained newer manual pages.
0 manual pages were added.
0 stray cats were added.
0 old database entries were purged.
```

Or run **makewhatis** on CentOS/Redhat.

```
[root@centos65 ~]# apropos scsi scsi: nothing
appropriate
[root@centos65 ~]# makewhatis  [root@centos65 ~]# apropos scsi hpsa
(4)  - HP Smart Array SCSI driver lsscsi               (8)  - list SCSI
devices (or hosts) and their attributes sd                   (4)  - Driver
for SCSI Disk Drives st                   (4)  - SCSI tape device
```

Working with directories

This module is a brief overview of the most common commands to work with directories: **pwd**, **cd**, **ls**, **mkdir** and **rmdir**. These commands are available on any Linux (or Unix) system.

This module also discusses **absolute** and **relative paths** and **path completion** in the **bash** shell.

pwd

The **you are here** sign can be displayed with the **pwd** command (Print Working Directory). Go ahead, try it: Open a command line interface (also called a terminal, console or xterm) and type **pwd**. The tool displays your **current directory**.

```
user@debian8:~$ pwd /home/user
```

cd

You can change your current directory with the **cd** command (Change Directory).

```
user@debian8$ cd /etc user@debian8$ pwd

/etc user@debian8$ cd /bin
user@debian8$ pwd
/bin user@debian8$ cd
/home/user/ user@debian8$
pwd /home/user
```

cd ~

The **cd** is also a shortcut to get back into your home directory. Just typing **cd** without a target directory, will put you in your home directory. Typing **cd ~** has the same effect.

```
user@debian8$ cd /etc
user@debian8$ pwd /etc
user@debian8$ cd
user@debian8$ pwd
/home/user user@debian8$
cd ~ user@debian8$ pwd

/home/user
```

cd ..

To go to the **parent directory** (the one just above your current directory in the directory tree), type **cd .. .**

```
user@debian8$ pwd
/usr/share/games user@debian8$
cd .. user@debian8$ pwd
/usr/share
```

*To stay in the current directory, type **cd .** ;-)* We will see useful use of the **.** character representing the current directory later.

cd -

Another useful shortcut with **cd** is to just type **cd -** to go to the previous directory.

```
user@debian8$ pwd

/home/user user@debian8$ cd /etc user@debian8$ pwd
/etc user@debian8$
cd -
/home/user user@debian8$ cd /etc
```

absolute and relative paths

You should be aware of **absolute and relative paths** in the file tree. When you type a path starting with a **slash** (/), then the **root** of the file tree is assumed. If you don't start your path with a slash, then the current directory is the assumed starting point.

The screenshot below first shows the current directory /**home**/**user**. From within this directory, you have to type **cd** /**home** instead of **cd home** to go to the /**home** directory.

```
user@debian8$ pwd
/home/user
user@debian8$ cd home
bash: cd: home: No such file or
directory user@debian8$ cd /home
user@debian8$ pwd
/home
```

When inside /**home**, you have to type **cd user** instead of **cd** /**user** to enter the subdirectory **user** of the current directory /**home**.

```
user@debian8$ pwd
/home
user@debian8$ cd /user
bash: cd: /user: No such file or directory user@debian8$ cd user user@debian8$
pwd /home/user
```

In case your current directory is the **root directory** /, then both **cd** /**home** and **cd home** will get you in the /**home** directory.

```
user@debian8$ pwd
/
user@debian8$ cd home user@debian8$ pwd /home user@debian8$ cd / user@debian8$ cd
/home user@debian8$ pwd /home
```

This was the last screenshot with **pwd** statements. From now on, the current directory will often be displayed in the prompt. Later in this book we will explain how the shell variable $**PS1** can be configured to show this.

path completion

The **tab key** can help you in typing a path without errors. Typing **cd** /**et** followed by the **tab key** will expand the command line to **cd** /**etc**/. When typing **cd** /**Et** followed by the **tab key**, nothing will happen because you typed the wrong **path** (upper case E).

You will need fewer key strokes when using the **tab key**, and you will be sure your typed **path** is correct!

ls

You can list the contents of a directory with **ls**.

```
user@debian8:~$ ls allfiles.txt  dmesg.txt  services   stuff
summer.txt user@debian8:~$
```

ls -a

A frequently used option with ls is -**a** to show all files. Showing all files means including the **hidden files**. When a file name on a Linux file system starts with a dot, it is considered a **hidden file** and it does not show up in regular file listings.

```
user@debian8:~$ ls

allfiles.txt  dmesg.txt  services  stuff  summer.txt user@debian8:~$
ls -a
.   allfiles.txt   .bash_profile  dmesg.txt   .lesshst  stuff
..  .bash_history  .bashrc        services    .ssh      summer.txt
user@debian8:~$
```

ls -l

Many times you will be using options with **ls** to display the contents of the directory in different formats or to display different parts of the directory. Typing just **ls** gives you a list of files in the directory. Typing **ls -l** (that is a letter L, not the number 1) gives you a long listing.

```
user@debian8:~$ ls -l total 17296
-rw-r--r-- 1 user user 17584442 Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 user user    96650 Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 user user    19558 Sep 17 00:04 services drwxr-xr-x 2 user user
4096 Sep 17 00:04 stuff
-rw-r--r-- 1 user user        0 Sep 17 00:04 summer.txt
```

ls -lh

Another frequently used ls option is **-h**. It shows the numbers (file sizes) in a more human readable format. Also shown below is some variation in the way you can give the options to **ls**. We will explain the details of the output later in this book.

Note that we use the letter L as an option in this screenshot, not the number 1.

```
user@debian8:~$ ls -l -h total 17M
-rw-r--r-- 1 user user  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 user user  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 user user  20K Sep 17 00:04 services drwxr-xr-x 2 user user 4.0K Sep 17 00:04
stuff
-rw-r--r-- 1 user user    0 Sep 17 00:04 summer.txt
user@debian8:~$ ls -lh total 17M
-rw-r--r-- 1 user user  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 user user  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 user user  20K Sep 17 00:04 services drwxr-xr-x 2 user user 4.0K Sep 17 00:04
stuff
-rw-r--r-- 1 user user    0 Sep 17 00:04 summer.txt
user@debian8:~$ ls -hl total 17M
-rw-r--r-- 1 user user  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 user user  95K Sep 17 00:03 dmesg.txt
-rw-r--r-- 1 user user  20K Sep 17 00:04 services drwxr-xr-x 2 user user 4.0K Sep 17 00:04
stuff
-rw-r--r-- 1 user user    0 Sep 17 00:04 summer.txt
user@debian8:~$ ls -h -l total 17M
-rw-r--r-- 1 user user  17M Sep 17 00:03 allfiles.txt
-rw-r--r-- 1 user user  95K Sep 17 00:03 dmesg.txt
```

```
-rw-r--r-- 1 user user  20K Sep 17 00:04 services drwxr-xr-x 2 user user 4.0K Sep 17 00:04
stuff
-rw-r--r-- 1 user user    0 Sep 17 00:04 summer.txt user@debian8:~$
```

mkdir

Walking around the Unix file tree is fun, but it is even more fun to create your own directories with **mkdir**. You have to give at least one parameter to **mkdir**, the name of the new directory to be created. Think before you type a leading / .

```
user@debian8:~$ mkdir mydir user@debian8:~$ cd mydir

user@debian8:~/mydir$ ls -al total 8 drwxr-xr-x  2
user user 4096 Sep 17 00:07 . drwxr-xr-x 48 user
user 4096 Sep 17 00:07 .. user@debian8:~/mydir$
mkdir stuff user@debian8:~/mydir$ mkdir otherstuff
user@debian8:~/mydir$ ls -l total 8
drwxr-xr-x 2 user user 4096 Sep 17 00:08 otherstuff drwxr-xrx 2 user
user 4096 Sep 17 00:08 stuff user@debian8:~/mydir$
```

mkdir -p

The following command will fail, because the **parent directory** of **threedirsdeep** does not exist.

```
user@debian8:~$ mkdir mydir2/mysubdir2/threedirsdeep
mkdir: cannot create directory `mydir2/mysubdir2/threedirsdeep': No such
fi\ le or directory
```

When given the option **-p**, then **mkdir** will create **parent directories** as needed.

rmdir

When a directory is empty, you can use **rmdir** to remove the directory.

```
user@debian8:~/mydir$ ls -l
total 8
drwxr-xr-x 2 user user 4096 Sep 17 00:08 otherstuff drwxr-
xrx 2 user user 4096 Sep 17 00:08 stuff user@debian8:~/mydir$
rmdir otherstuff user@debian8:~/mydir$  cd     ..

user@debian8:~$ rmdir mydir
rmdir: failed to remove `mydir': Directory not empty
user@debian8:~$         rmdir         mydir/stuff

user@debian8:~$ rmdir mydir
```

rmdir -p

And similar to the **mkdir -p** option, you can also use **rmdir** to recursively remove directories.

```
user@debian8:~$ mkdir -p test42/subdir user@debian8:~$ rmdir -p

test42/subdir
```

working with files

In this chapter we learn how to recognise, create, remove, copy and move files using commands like **file**, **touch**, **rm**, **cp**, **mv** and **rename**.

all files are case sensitive

Files on Linux (or any Unix) are **case sensitive**. This means that **FILE1** is different from **file1**, and **/etc/hosts** is different from **/etc/Hosts** (the latter one does not exist on a typical Linux computer).

This screenshot shows the difference between two files, one with upper case **W**, the other with lower case **w**.

```
user@predator-pc~/Linux$       ls
```

```
winter.txt  Winter.txt user@predator-
pc~/Linux$       cat winter.txt
It is cold. user@predatorpc~/Linux$ cat
Winter.txt It is very cold!
```

Everything is a file

A **directory** is a special kind of **file**, but it is still a (case sensitive!) **file**. Each terminal window (for example /**dev**/**pts**/**4**), any hard disk or partition (for example /**dev**/**sdb1**) and any process are all represented somewhere in the **file system** as a **file**. It will become clear throughout this course that everything on Linux is a **file**.

file

The **file** utility determines the file type. Linux does not use extensions to determine the file type. The command line does not care whether a file ends in .txt or .pdf. As a system administrator, you should use the **file** command to determine the file type. Here are some examples on a typical Linux system.

```
user@predator-pc~$ file pic33.png
pic33.png: PNG image data, 3840 x 1200, 8-bit/color RGBA, non-interlaced
user@predator-pc~$ file /etc/passwd
/etc/passwd: ASCII text user@predator-pc~$ file HelloWorld.c HelloWorld.c: ASCII
C program text
```

The file command uses a magic file that contains patterns to recognise file types. The magic file is located in /**usr**/**share**/**file**/**magic**. Type **man 5 magic** for more information. It is interesting to point out **file** -**s** for special files like those in /**dev** and /**proc**.

```
root@debian6~# file /dev/sda
/dev/sda:        block        special

root@debian6~# file -s /dev/sda
/dev/sda: x86 boot sector; partition 1: ID=0x83, active, starthead...

root@debian6~# file /proc/cpuinfo
/proc/cpuinfo:   empty   root@debian6~#   file   -s
/proc/cpuinfo /proc/cpuinfo:

ASCII C++ program text
```

touch create an empty file

One easy way to create an empty file is with **touch**. (We will see many other ways for creating files later in this book.)

This screenshot starts with an empty directory, creates two files with **touch** and the lists those files.

```
user@debian7:~$ ls -l
total 0
user@debian7:~$ touch file42 user@debian7:~$ touch file33 user@debian7:~$ ls l
total 0
-rw-r--r-- 1 user user 0 Oct 15 08:57 file33 -rw-r--r-- 1 user user 0 Oct 15
08:56 file42 user@debian7:~$
```

touch -t

The **touch** command can set some properties while creating empty files. Can you determine what is set by looking at the next screenshot? If not, check the manual for **touch**.

```
user@debian7:~$ touch -t 200505050000 SinkoDeMayo user@debian7:~$ touch -t
130207111630 BigBattle.txt
user@debian7:~$ ls -l total
0
-rw-r--r-- 1 user user 0 Jul 11  1302 BigBattle.txt
-rw-r--r-- 1 user user 0 Oct 15 08:57 file33
-rw-r--r-- 1 user user 0 Oct 15 08:56 file42 -rw-r--r--
1 user user 0 May  5  2005 SinkoDeMayo user@debian7:~$
```

rm

remove forever

When you no longer need a file, use **rm** to remove it. Unlike some graphical user interfaces, the command line in general does not have a **waste bin** or **trash can** to recover files. When you use **rm** to remove a file, the file is gone. Therefore, be careful when removing files!

```
user@debian7:~$ ls
BigBattle.txt  file33  file42  SinkoDeMayo user@debian7:~$ rm
BigBattle.txt user@debian7:~$ ls file33  file42  SinkoDeMayo
user@debian7:~$
```

rm -i

To prevent yourself from accidentally removing a file, you can type **rm -i**.

```
user@debian7:~$ ls file33  file42
             SinkoDeMayo
user@debian7:~$ rm -i file33
rm: remove regular empty file `file33'? yes
user@debian7:~$ rm -i SinkoDeMayo
rm: remove regular empty file `SinkoDeMayo'? n
user@debian7:~$   ls   file42      SinkoDeMayo
user@debian7:~$
```

rm -rf

By default, **rm -r** will not remove non-empty directories. However, **rm** accepts several options that will allow you to remove any directory. The **rm -rf** statement is famous because it will erase anything (providing that you have the permissions to do so). When you are logged on as root, be very careful with **rm -rf** (the **f** means **force** and the **r** means **recursive**) since being root implies that permissions don't apply to you. You can literally erase your entire file system by accident.

```
user@debian7:~$        mkdir        test
user@debian7:~$ rm test
rm: cannot remove `test': Is a directory
user@debian7:~$ rm -rf test user@debian7:~$
ls test
ls: cannot access test: No such file or directory
user@debian7:~$
```

cp copy one file

To copy a file, use **cp** with a source and a target argument.

```
user@debian7:~$       ls
file42  SinkoDeMayo
user@debian7:~$ cp file42 file42.copy user@debian7:~$ ls
file42  file42.copy  SinkoDeMayo
```

copy to another directory

If the target is a directory, then the source files are copied to that target directory.

```
user@debian7:~$  mkdir  dir42 user@debian7:~$ cp SinkoDeMayo dir42
user@debian7:~$ ls dir42/ SinkoDeMayo
```

cp -r

To copy complete directories, use **cp -r** (the **-r** option forces **recursive** copying of all files in all subdirectories).

```
user@debian7:~$ ls dir42  file42  file42.copy  SinkoDeMayo user@debian7:~$
cp -r dir42/ dir33 user@debian7:~$ ls dir33  dir42  file42  file42.copy
SinkoDeMayo user@debian7:~$ ls dir33/ SinkoDeMayo
```

copy multiple files to directory

You can also use cp to copy multiple files into a directory. In this case, the last argument (a.k.a. the target) must be a directory.

```
user@debian7:~$ cp file42 file42.copy SinkoDeMayo dir42/
user@debian7:~$ ls dir42/ file42  file42.copy  SinkoDeMayo
```

cp -i

To prevent **cp** from overwriting existing files, use the **-i** (for interactive) option.

```
  user@debian7:~$     cp     SinkoDeMayo   file42
  user@debian7:~$     cp     SinkoDeMayo   file42

user@debian7:~$ cp -i SinkoDeMayo file42
cp: overwrite `file42'? n user@debian7:~$
```

mv

rename files with mv

Use **mv** to rename a file or to move the file to another directory.

```
user@debian7:~$ ls dir33  dir42  file42  file42.copy  SinkoDeMayo
user@debian7:~$ mv file42 file33 user@debian7:~$ ls
dir33  dir42  file33  file42.copy  SinkoDeMayo user@debian7:~$
```

When you need to rename only one file then **mv** is the preferred command to use.

rename directories with mv

The same **mv** command can be used to rename directories.

```
user@debian7:~$ ls -l
total 8
drwxr-xr-x 2 user user 4096 Oct 15 09:36 dir33 drwxr-xr-x 2 user
user 4096 Oct 15 09:36 dir42 -rw-r--r-- 1 user user     0 Oct 15
09:38 file33
-rw-r--r-- 1 user user     0 Oct 15 09:16 file42.copy -rw-r--r-- 1
user user     0 May  5  2005 SinkoDeMayo
user@debian7:~$ mv dir33 backup user@debian7:~$ ls -l
total 8
drwxr-xr-x 2 user user 4096 Oct 15 09:36 backup drwxr-xr-x 2 user
user 4096 Oct 15 09:36 dir42 -rw-r--r-- 1 user user     0 Oct 15
09:38 file33
-rw-r--r-- 1 user user     0 Oct 15 09:16 file42.copy -rw-r--r-- 1
user user     0 May  5  2005 SinkoDeMayo user@debian7:~$
```

mv -i

The **mv** also has a **-i** switch similar to **cp** and **rm**. this screenshot shows that **mv -i** will ask permission to overwrite an existing file.

```
user@debian7:~$ mv -i file33 SinkoDeMayo mv: overwrite

`SinkoDeMayo'? no
user@debian7:~$
```

Rename

The **rename** command is one of the rare occasions where the Linux Fundamentals book has to make a distinction between Linux distributions. Almost every command in the **Fundamentals** part of this book works on almost every Linux computer. But **rename** is different.

Try to use **mv** whenever you need to rename only a couple of files.

Rename on Debian/Debian stable

The **rename** command on Debian uses regular expressions (regular expression or shor regex are explained in a later chapter) to rename many files at once.

Below a **rename** example that switches all occurrences of txt to png for all file names ending in .txt.

```
user@debian7:~/test42$  ls abc.txt  file33.txt
file42.txt user@debian7:~/test42$ rename
's/\.txt/\.png/' *.txt user@debian7:~/test42$
   ls abc.png  file33.png  file42.png
```

This second example switches all (first) occurrences of **file** into **document** for all file names ending in .png.

```
user@debian7:~/test42$  ls abc.png  file33.png
file42.png user@debian7:~/test42$ rename
's/file/document/' *.png user@debian7:~/test42$ ls
abc.png  document33.png  document42.png
user@debian7:~/test42$
```

Rename on CentOS/RHEL/Fedora

On Red Hat Enterprise Linux, the syntax of **rename** is a bit different. The first example below renames all *.conf files replacing any occurrence of .conf with .backup.

```
[user@centos7 ~]$ touch one.conf two.conf three.conf
[user@centos7 ~]$ rename .conf .backup *.conf
[user@centos7 ~]$ ls
one.backup  three.backup  two.backup [user@centos7 ~]$
```

The second example renames all (*) files replacing one with ONE.

```
[user@centos7 ~]$ ls
one.backup  three.backup  two.backup [user@centos7 ~]$ rename one ONE *
[user@centos7 ~]$ ls
ONE.backup  three.backup  two.backup
[user@centos7 ~]$
```

Head

You can use **head** to display the first ten lines of a file.

```
user@debian7~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh        sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh root@debian7~#
```

The **head** command can also display the first **n** lines of a file.

```
user@debian7~$    head   -4     /etc/passwd
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh user@debian7~$
```

And **head** can also display the first **n bytes**.

```
user@debian7~$ head -c14 /etc/passwd root:x:0:0:roouser@debian7~$
```

tail

Similar to **head**, the **tail** command will display the last ten lines of a file.

```
user@debian7~$ tail /etc/services vboxd            20012/udp binkp
24554/tcp                       # binkp fidonet protocol asp

27374/tcp                       # Address Search Protocol
asp             27374/udp
csync2          30865/tcp                       # cluster synchronization tool
dircproxy       57000/tcp                         # Detachable IRC Proxy tfido
60177/tcp                       # fidonet EMSI over telnet fido      60179/tcp
# fidonet EMSI over TCP

#         Local         services

user@debian7~$
```

You can give **tail** the number of lines you want to see.

```
user@debian7~$ tail -3 /etc/services fido
60179/tcp                       # fidonet EMSI over TCP
```

```
#   Local  services user@debian7~$
```

The **tail** command has other useful options, some of which we will use during this course.

cat

The **cat** command is one of the most universal tools, yet all it does is copy **standard input** to **standard output**. In combination with the shell this can be very powerful and diverse. Some examples will give a glimpse into the possibilities. The first example is simple, you can use cat to display a file on the screen. If the file is longer than the screen, it will scroll to the end.

```
user@debian8:~$ cat /etc/resolv.conf
domain     linux-training.be search
    linux-training.be nameserver
192.168.1.42
```

concatenate

cat is short for **concatenate**. One of the basic uses of **cat** is to concatenate files into a bigger (or complete) file.

```
user@debian8:~$ echo one >part1 user@debian8:~$
echo two >part2 user@debian8:~$ echo three

>part3
user@debian8:~$ cat part1 one
user@debian8:~$ cat part2 two
user@debian8:~$ cat part3 three
user@debian8:~$ cat part1 part2
part3 one two three
user@debian8:~$ cat part1 part2 part3 >all
user@debian8:~$  cat  all  one  two  three
user@debian8:~$
```

create files

You can use **cat** to create flat text files. Type the **cat > winter.txt** command as shown in the screenshot. Then type one or more lines, finishing each line with the enter key. After the last line, type and hold the Control (Ctrl) key and press d.

```
user@debian8:~$ cat > winter.txt
It is very cold today!
user@debian8:~$ cat winter.txt
It is very cold today! user@debian8:~$
```

The **Ctrl d** key combination will send an **EOF** (End of File) to the running process ending the **cat** command.

custom end marker

You can choose an end marker for **cat** with << as is shown in this screenshot. This construction is called a **here directive** and will end the **cat** command.

```
user@debian8:~$ cat > hot.txt <<stop >
It is hot today!
> Yes it is summer.
> stop
user@debian8:~$  cat  hot.txt  It
is hot today!
Yes it is summer. user@debian8:~$
```

copy files

In the third example you will see that cat can be used to copy files. We will explain in detail what happens here in the bash shell chapter.

```
user@debian8:~$ cat winter.txt
It is very cold today! user@debian8:~$ cat winter.txt
> cold.txt user@debian8:~$ cat cold.txt
It is very cold today! user@debian8:~$
```

tac

Just one example will show you the purpose of **tac** (cat backwards).

```
user@debian8:~$ cat count
one two three four
user@debian8:~$ tac count
four three two one
```

cat

When between two **pipes**, the **cat** command does nothing (except putting **stdin** on **stdout**).

```
[user@RHEL4b pipes]$ tac count.txt | cat | cat | cat | cat | cat five
four three two one
[user@RHEL4b pipes]$
```

tee

Writing long **pipes** in Unix is fun, but sometimes you may want intermediate results. This is were **tee** comes in handy. The **tee** filter puts **stdin** on **stdout** and also into a file. So **tee** is almost the same as **cat**, except that it has two identical outputs.

```
[user@RHEL4b pipes]$ tac count.txt | tee temp.txt | tac
one two three four five
[user@RHEL4b pipes]$ cat temp.txt
five four three two one
[user@RHEL4b pipes]$
```

grep

The **grep** filter is famous among Unix users. The most common use of **grep** is to filter lines of text containing (or not containing) a certain string.

```
[user@RHEL4b pipes]$ cat tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[user@RHEL4b pipes]$ cat tennis.txt | grep Williams
Serena Williams, usa
Venus Williams, USA
```

You can write this without the cat.

```
[user@RHEL4b pipes]$ grep Williams tennis.txt
Serena Williams, usa
Venus Williams, USA
```

One of the most useful options of grep is **grep -i** which filters in a case insensitive way.

```
[user@RHEL4b pipes]$ grep Bel tennis.txt
Justine Henin, Bel
[user@RHEL4b pipes]$ grep -i Bel tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
[user@RHEL4b pipes]$
```

Another very useful option is **grep -v** which outputs lines not matching the string.

```
[user@RHEL4b pipes]$ grep -v Fra tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[user@RHEL4b pipes]$
```

And of course, both options can be combined to filter all lines not containing a case insensitive string.

```
[user@RHEL4b pipes]$ grep -vi usa tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
[user@RHEL4b pipes]$
```

With **grep -A1** one line **after** the result is also displayed.

```
user@debian5:~/pipes$ grep -A1 Henin tennis.txt
Justine Henin, Bel
Serena Williams, usa
```

With **grep -B1** one line **before** the result is also displayed.

```
user@debian5:~/pipes$ grep -B1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
```

With **grep -C1** (context) one line **before** and one **after** are also displayed. All three options (A,B, and C) can display any number of lines (using e.g. A2, B4 or C20).

```
user@debian5:~/pipes$ grep -C1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
```

WC

Counting words, lines and characters is easy with **wc**.

```
[user@RHEL4b pipes]$ wc tennis.txt
  5  15 100 tennis.txt
[user@RHEL4b pipes]$ wc -l tennis.txt  5 tennis.txt
[user@RHEL4b pipes]$ wc -w tennis.txt  15 tennis.txt
[user@RHEL4b pipes]$ wc -c tennis.txt
100 tennis.txt
[user@RHEL4b pipes]$
```

sort

The **sort** filter will default to an alphabetical sort.

```
user@debian5:~/pipes$ cat music.txt
Queen
Brel
Led Zeppelin Abba user@debian5:~/pipes$ sort music.txt
Abba
Brel
Led Zeppelin
Queen
```

But the **sort** filter has many options to tweak its usage. This example shows sorting different columns (column 1 or column 2).

```
[user@RHEL4b pipes]$ sort -k1 country.txt

Belgium, Brussels, 10
France, Paris, 60
Germany, Berlin, 100
Iran, Teheran, 70
Italy, Rome, 50
[user@RHEL4b pipes]$ sort -k2 country.txt
Germany, Berlin, 100
Belgium, Brussels, 10
France, Paris, 60
Italy, Rome, 50
Iran, Teheran, 70
```

The screenshot below shows the difference between an alphabetical sort and a numerical sort (both on the third column).

```
[user@RHEL4b pipes]$ sort -k3 country.txt

Belgium, Brussels, 10
Germany, Berlin, 100 Italy,
Rome, 50
France, Paris, 60
Iran, Teheran, 70
[user@RHEL4b pipes]$ sort -n -k3 country.txt
Belgium, Brussels, 10
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
Germany, Berlin, 100
```

uniq

With **uniq** you can remove duplicates from a **sorted list**.

```
user@debian5:~/pipes$ cat music.txt

Queen
Brel
Queen Abba user@debian5:~/pipes$
sort music.txt
Abba
Brel Queen Queen user@debian5:~/pipes$
sort music.txt |uniq
Abba
Brel
Queen
```

find

The **find** command can be very useful at the start of a pipe to search for files. Here are some examples. You might want to add **2**>/**dev**/**null** to the command lines to avoid cluttering your screen with error messages.

Find all files in /**etc** and put the list in etcfiles.txt

```
find /etc > etcfiles.txt
```

Find all files of the entire system and put the list in allfiles.txt

```
find / > allfiles.txt
```

Find files that end in .conf in the current directory (and all subdirs).

```
find . -name "*.conf"
```

Find files of type file (not directory, pipe or etc.) that end in .conf.

```
find . -type f -name "*.conf"
```

Find files of type directory that end in .bak .

```
find /data -type d -name "*.bak"
```

Find files that are newer than file42.txt

```
find . -newer file42.txt
```

Locate

The **locate** tool is very different from **find** in that it uses an index to locate files. This is a lot faster than traversing all the directories, but it also means that it is always outdated. If the index

does not exist yet, then you have to create it (as root on Red Hat Enterprise Linux) with the **updatedb** command.

```
[user@RHEL4b ~]$ locate Samba warning: locate: could not open database:
/var/lib/slocate/slocate.db:... warning: You need to run the 'updatedb' command
(as root) to create th...
Please have a look at /etc/updatedb.conf to enable the daily cron job.
[user@RHEL4b ~]$ updatedb fatal error: updatedb: You are not authorized to create
a default sloc... [user@RHEL4b ~]$ su Password:
[root@RHEL4b ~]# updatedb
[root@RHEL4b ~]#
```

Most Linux distributions will schedule the **updatedb** to run once every day.

date

The **date** command can display the date, time, time zone and more.

```
user@rhel55 ~$ date Sat Apr 17

12:44:30 CEST 2010
```

A date string can be customised to display the format of your choice. Check the man page for more options.

```
user@rhel55 ~$ date +'%A %d-%m-%Y' Saturday

17-04-2010
```

Time on any Unix is calculated in number of seconds since 1969 (the first second being the first second of the first of January 1970). Use **date +%s** to display Unix time in seconds.

```
user@rhel55 ~$ date +%s
1271501080
```

When will this seconds counter reach two thousand million ?

```
user@rhel55 ~$ date -d '1970-01-01 + 2000000000 seconds'

Wed May 18 04:33:20 CEST 2033
```

cal

The **cal** command displays the current month, with the current day highlighted.

```
user@rhel55 ~$ cal      April 2010
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

You can select any month in the past or the future.

```
user@rhel55 ~$ cal 2 1970
   February 1970
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
```

```
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
```

sleep

The **sleep** command is sometimes used in scripts to wait a number of seconds. This example shows a five second **sleep**.

```
user@rhel55 ~$ sleep 5 user@rhel55 ~$
```

time

The **time** command can display how long it takes to execute a command. The **date** command takes only a little time.

```
user@rhel55 ~$ time date Sat
Apr 17 13:08:27 CEST 2010

real    0m0.014s
user    0m0.008s
sys     0m0.006s
```

The **sleep 5** command takes five **real** seconds to execute, but consumes little **cpu time**.

```
user@rhel55 ~$ time sleep
5

real    0m5.018s
user    0m0.005s
sys     0m0.011s
```

This **bzip2** command compresses a file and uses a lot of **cpu time**.

```
user@rhel55 ~$ time bzip2 text.txt

real    0m2.368s
user    0m0.847s
sys     0m0.539s
```

gzip - gunzip

Users never have enough disk space, so compression comes in handy. The **gzip** command can make files take up less space.

```
user@rhel55 ~$ ls -lh text.txt
-rw-rw-r-- 1 user user 6.4M Apr 17 13:11 text.txt user@rhel55 ~$ gzip text.txt
user@rhel55

~$ ls -lh text.txt.gz
-rw-rw-r-- 1 user user 760K Apr 17 13:11 text.txt.gz
```

You can get the original back with **gunzip**.

```
user@rhel55 ~$ gunzip text.txt.gz user@rhel55

~$ ls -lh text.txt
-rw-rw-r-- 1 user user 6.4M Apr 17 13:11 text.txt
```

repeating the last command

To repeat the last command in bash, type !!. This is pronounced as **bang bang**.

```
user@debian5:~/test42$ echo this will be repeated > file42.txt
user@debian5:~/test42$ !! echo this will be repeated > file42.txt
user@debian5:~/test42$
```

repeating other commands

You can repeat other commands using one **bang** followed by one or more characters. The shell will repeat the last command that started with those characters.

```
user@debian5:~/test42$ touch file42
user@debian5:~/test42$ cat file42
user@debian5:~/test42$ !to touch
file42 user@debian5:~/test42$
```

history

To see older commands, use **history** to display the shell command history (or use **history n** to see the last n commands).

```
user@debian5:~/test$ history 10
38  mkdir test
39  cd test
40  touch file1
41  echo hello > file2
42  echo It is very cold today > winter.txt
43  ls
44  ls -l
45  cp winter.txt summer.txt
46  ls -l
47  history 10
```

!n

When typing ! followed by the number preceding the command you want repeated, then the shell will echo the command and execute it.

```
user@debian5:~/test$ !43
ls file1  file2  summer.txt
winter.txt
```

Ctrl-r

Another option is to use **ctrl**-**r** to search in the history. In the screenshot below i only typed **ctrl**-**r** followed by four characters **apti** and it finds the last command containing these four consecutive characters.

```
user@debian5:~$ (reverse-i-search)`apti': sudo aptitude

install screen
```

HISTSIZE

The $HISTSIZE variable determines the number of commands that will be remembered in your current environment. Most distributions default this variable to 500 or 1000.

```
user@debian5:~$ echo $HISTSIZE 500
```

You can change it to any value you like.

```
user@debian5:~$  HISTSIZE=15000 user@debian5:~$ echo
$HISTSIZE 15000
```

HISTFILE

The $HISTFILE variable points to the file that contains your history. The **bash** shell defaults this value to ~/**.bash_history**.

```
user@debian5:~$ echo $HISTFILE /home/user/.bash_history
```

A session history is saved to this file when you **exit** the session!

*Closing a gnome-terminal with the mouse, or typing **reboot** as root will NOT save your terminal's history.*

HISTFILESIZE

The number of commands kept in your history file can be set using $HISTFILESIZE.

```
user@debian5:~$ echo $HISTFILESIZE 15000
```

prevent recording a command

You can prevent a command from being recorded in **history** using a space prefix.

```
user@debian8:~/github$ echo abc abc
user@debian8:~/github$  echo def def
user@debian8:~/github$ echo ghi ghi
user@debian8:~/github$ history 3
9501  echo abc
9502  echo ghi
 9503  history 3
```

(optional)regular expressions

It is possible to use **regular expressions** when using the **bang** to repeat commands. The screenshot below switches 1 into 2.

```
user@debian5:~/test$           cat           file1

user@debian5:~/test$ !c:s/1/2 cat
       file2

hello
user@debian5:~/test$
```

(optional) Korn shell history

Repeating a command in the **Korn shell** is very similar. The Korn shell also has the **history** command, but uses the letter **r** to recall lines from history.

This screenshot shows the history command. Note the different meaning of the parameter.

```
$ history 17
```

```
17  clear
18  echo hoi
19  history 12
20  echo world
21  history 17
```

Repeating with **r** can be combined with the line numbers given by the history command, or with the first few letters of the command.

```
$ r e echo
world world
$ cd /etc
$ r cd /etc
$
```

Introduction to users

This little chapter will teach you how to identify your user account on a Unix computer using commands like **who am i**, **id**, and more.

In a second part you will learn how to become another user with the **su** command.

And you will learn how to run a program as another user with **sudo**.

whoami

The **whoami** command tells you your username.

```
[user@centos7 ~]$ whoami user
[user@centos7 ~]$
```

who

The **who** command will give you information about who is logged on the system.

```
[user@centos7 ~]$ who root     pts/0        2014-10-10 23:07
(10.104.33.101) user     pts/1
2014-10-10 23:30 (10.104.33.101) laura    pts/2        2014-10-10 23:34
(10.104.33.96) tania     pts/3         2014-10-10 23:39 (10.104.33.91)
[user@centos7 ~]$
```

who am i

With **who am i** the **who** command will display only the line pointing to your current session.

```
[user@centos7 ~]$ who am i
user     pts/1        2014-10-10 23:30 (10.104.33.101) [user@centos7 ~]$
```

w

The **w** command shows you who is logged on and what they are doing.

```
[user@centos7 ~]$ w

 23:34:07 up 31 min,  2 users,  load average: 0.00, 0.01, 0.02
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT root
pts/0     23:07   15.00s  0.01s  0.01s top user     pts/1
23:30    7.00s  0.00s  0.00s w [user@centos7 ~]$
```

id

The **id** command will give you your user id, primary group id, and a list of the groups that you belong to.

```
user@debian7:~$ id uid=1000(user) gid=1000(user) groups=1000(user)
```

On RHEL/CentOS you will also get **SELinux** context information with this command.

```
[root@centos7 ~]# id uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r\ :unconfined_t:s0-s0:c0.c1023
```

su to another user

The **su** command allows a user to run a shell as another user.

```
laura@debian7:~$ su tania Password:
tania@debian7:/home/laura$
```

su to root

Yes you can also **su** to become **root**, when you know the **root password**.

```
laura@debian7:~$ su root Password:
root@debian7:/home/laura#
```

su as root

You need to know the password of the user you want to substitute to, unless your are logged in as **root**. The **root** user can become any existing user without knowing that user's password.

```
root@debian7:~# id uid=0(root) gid=0(root) groups=0(root) root@debian7:~#
su -

valentina valentina@debian7:~$
```

su - $username

By default, the **su** command maintains the same shell environment. To become another user and also get the target user's environment, issue the **su -** command followed by the target username.

```
root@debian7:~# su laura laura@debian7:/root$

exit
exit root@debian7:~# su - laura laura@debian7:~$ pwd
/home/laura
```

su -

When no username is provided to **su** or **su -**, the command will assume **root** is the target.

```
tania@debian7:~$ su -
Password:
root@debian7:~#
```

run a program as another user

The sudo program allows a user to start a program with the credentials of another user. Before this works, the system administrator has to set up the **/etc/sudoers** file. This can be useful to delegate administrative tasks to another user (without giving the root password).

The screenshot below shows the usage of **sudo**. User **user** received the right to run **useradd** with the credentials of **root**. This allows **user** to create new users on the system without becoming **root** and without knowing the **root password**.

First the command fails for **user**.

```
user@debian7:~$ /usr/sbin/useradd -m valentina useradd:
Permission denied. useradd: cannot lock /etc/passwd; try
again later.
```

But with **sudo** it works.

```
user@debian7:~$ sudo /usr/sbin/useradd -m valentina
[sudo] password for user: user@debian7:~$
```

visudo

Check the man page of **visudo** before playing with the **/etc/sudoers** file. Editing the **sudoers** is out of scope for this fundamentals book.

```
user@rhel65:~$ apropos visudo
visudo               (8)  - edit the sudoers file
user@rhel65:~$
```

sudo su -

On some Linux systems like Debian stable and XDebian stable, the **root** user does not have a password set. This means that it is not possible to login as **root** (extra security). To perform tasks as **root**, the first user is given all **sudo rights** via the **/etc/sudoers**. In fact all users that are members of the admin group can use sudo to run all commands as root.

```
root@predator-pc~# grep admin /etc/sudoers
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

The end result of this is that the user can type **sudo su -** and become root without having to enter the root password. The sudo command does require you to enter your own password. Thus the password prompt in the screenshot below is for sudo, not for su.

```
user@predator-pc~$ sudo su - Password:
root@predator-pc~#
```

sudo logging

Using **sudo** without authorization will result in a severe warning:

```
user@rhel65:~$ sudo su -
We trust you have received the usual lecture from the local System Administrator.
It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.
[sudo] password for user: user is not in the sudoers file.  This incident will be
reported. user@rhel65:~$
```

The root user can see this in the **/var/log/secure** on Red Hat and in **/var/log/auth.log** on Debian).

```
root@rhel65:~# tail /var/log/secure | grep sudo | tr -s ' '
Apr 13 16:03:42 rhel65 sudo: user : user NOT in sudoers ; TTY=pts/0 ; PWD=\
/home/user ; USER=root ; COMMAND=/bin/su - root@rhel65:~#
```

user management

This chapter will teach you how to use **useradd**, **usermod** and **userdel** to create, modify and remove user accounts. You will need **root** access on a Linux computer to complete this chapter.

User management on Linux can be done in three complementary ways. You can use the **graphical** tools provided by your distribution. These tools have a look and feel that depends on the distribution. If you are a novice Linux user on your home system, then use the graphical tool that is provided by your distribution. This will make sure that you do not run into problems.

Another option is to use **command line tools** like useradd, usermod, gpasswd, passwd and others. Server administrators are likely to use these tools, since they are familiar and very similar across many different distributions. This chapter will focus on these command line tools.

A third and rather extremist way is to **edit the local configuration files** directly using vi (or vipw/vigr). Do not attempt this as a novice on production systems!

/etc/passwd

The local user database on Linux (and on most Unixes) is **/etc/passwd**.

```
[root@RHEL5    ~]#    tail    /etc/passwd    inge:x:518:524:art
dealer:/home/inge:/bin/ksh                ann:x:519:525:flute
player:/home/ann:/bin/bash          frederik:x:520:526:rubius
poet:/home/frederik:/bin/bash          steven:x:521:527:roman
emperor:/home/steven:/bin/bash
pascale:x:522:528:artist:/home/pascale:/bin/ksh
geert:x:524:530:kernel       developer:/home/geert:/bin/bash
wim:x:525:531:master               damuti:/home/wim:/bin/bash
sandra:x:526:532:radish stresser:/home/sandra:/bin/bash
```

As you can see, this file contains seven columns separated by a colon. The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory, and the login shell.

More information can be found by typing **man 5 passwd**.

```
[root@RHEL5 ~]# man 5 passwd
```

root

The **root** user also called the **superuser** is the most powerful account on your Linux system. This user can do almost anything, including the creation of other users. The root user always has userid 0 (regardless of the name of the account).

```
[root@RHEL5 ~]# head -1 /etc/passwd root:x:0:0:root:/root:/bin/bash
```

Useradd

You can add users with the **useradd** command. The example below shows how to add a user named yanina (last parameter) and at the same time forcing the creation of the home directory (-m), setting the name of the home directory (-d), and setting a description (-c).

```
[root@RHEL5 ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina
[root@RHEL5 ~]# tail -1 /etc/passwd yanina:x:529:529:yanina
wickmayer:/home/yanina:/bin/bash
```

The user named yanina received userid 529 and **primary group** id 529.

/etc/default/useradd

Both Red Hat Enterprise Linux and Debian/Debian stable have a file called **/etc/default/useradd** that contains some default user options. Besides using cat to display this file, you can also use **useradd -D**.

```
[root@RHEL4 ~]# useradd -D

GROUP=100
HOME=/home INACTIVE=-
1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

userdel

You can delete the user yanina with **userdel**. The -r option of userdel will also remove the home directory.

```
[root@RHEL5 ~]# userdel -r yanina
```

usermod

You can modify the properties of a user with the **usermod** command. This example uses **usermod** to change the description of the user harry.

```
[root@RHEL4 ~]# tail -1 /etc/passwd harry:x:516:520:harry
potter:/home/harry:/bin/bash
[root@RHEL4 ~]# usermod -c 'wizard' harry
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:wizard:/home/harry:/bin/bash
```

creating home directories

The easiest way to create a home directory is to supply the **-m** option with **useradd** (it is likely set as a default option on Linux).

A less easy way is to create a home directory manually with **mkdir** which also requires setting the owner and the permissions on the directory with **chmod** and **chown** (both commands are discussed in detail in another chapter).

```
[root@RHEL5 ~]# mkdir /home/laura

[root@RHEL5 ~]# chown laura:laura /home/laura
[root@RHEL5 ~]# chmod 700 /home/laura [root@RHEL5
~]# ls -ld /home/laura/
drwx------ 2 laura laura 4096 Jun 24 15:17 /home/laura/
```

/etc/skel/

When using **useradd** the **-m** option, the **/etc/skel/** directory is copied to the newly created home directory. The **/etc/skel/** directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way **/etc/skel/** serves as a default home directory and as a default user profile.

```
[root@RHEL5 ~]# ls -la /etc/skel/ total 48
drwxr-xr-x  2 root root  4096 Apr  1 00:11 .
drwxr-xr-x 97 root root 12288 Jun 24 15:36 ..
-rw-r--r--  1 root root    24 Jul 12  2006 .bash_logout
-rw-r--r--  1 root root   176 Jul 12  2006 .bash_profile
-rw-r--r--  1 root root   124 Jul 12  2006 .bashrc
```

deleting home directories

The -r option of **userdel** will make sure that the home directory is deleted together with the user account.

```
[root@RHEL5 ~]# ls -ld /home/wim/ drwx------ 2 wim wim 4096 Jun 24
15:19 /home/wim/
[root@RHEL5 ~]# userdel -r wim
[root@RHEL5 ~]# ls -ld /home/wim/ ls:

/home/wim/: No such file or directory
```

login shell

The **/etc/passwd** file specifies the **login shell** for the user. In the screenshot below you can see that user annelies will log in with the **/bin/bash** shell, and user laura with the **/bin/ksh** shell.

```
[root@RHEL5 ~]# tail -2 /etc/passwd annelies:x:527:533:sword
fighter:/home/annelies:/bin/bash laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

You can use the usermod command to change the shell for a user.

```
[root@RHEL5 ~]# usermod -s /bin/bash laura
[root@RHEL5 ~]#    tail   -1     /etc/passwd        laura:x:528:534:art
dealer:/home/laura:/bin/bash
```

chsh

Users can change their login shell with the **chsh** command. First, user harry obtains a list of available shells (he could also have done a **cat /etc/shells**) and then changes his login shell to the **Korn shell** (/bin/ksh). At the next login, harry will default into ksh instead of bash.

```
[laura@centos7 ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin /usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/bin/ksh
/bin/tcsh
/bin/csh
[laura@centos7 ~]$
```

Note that the **-l** option does not exist on Debian and that the above screenshot assumes that **ksh** and **csh** shells are installed.

The screenshot below shows how **laura** can change her default shell (active on next login).

```
[laura@centos7 ~]$ chsh -s /bin/ksh Changing shell for laura.
Password:
Shell changed.
```

user passwords

This chapter will tell you more about passwords for local users.
Three methods for setting passwords are explained; using the **passwd** command, using **openssel passwd**, and using the **crypt** function in a C program.
The chapter will also discuss password settings and disabling, suspending or locking accounts.

passwd

Passwords of users can be set with the **passwd** command. Users will have to provide their old password before twice entering the new one.

```
[tania@centos7     ~]$     passwd

Changing password for user tania.
Changing password for tania.
(current) UNIX password:
New password:
BAD PASSWORD: The password is shorter than 8 characters New
password:
BAD PASSWORD: The password is a palindrome New
password:
BAD PASSWORD: The password is too similar to the old one passwd:

Have exhausted maximum number of retries for service
```

As you can see, the passwd tool will do some basic verification to prevent users from using too simple passwords. The **root** user does not have to follow these rules (there will be a warning though). The **root** user also does not have to provide the old password before entering the new password twice.

```
root@debian7:~# passwd tania
Enter new UNIX password: Retype
new UNIX password: passwd:
password updated successfully
```

shadow file

User passwords are encrypted and kept in **/etc/shadow**. The /etc/shadow file is read only and can only be read by root. We will see in the file permissions section how it is possible for users to change their password. For now, you will have to know that users can change their password with the **/usr/bin/passwd** command.

```
[root@centos7 ~]# tail -4 /etc/shadow
user:$6$ikp2Xta5BT.Tml.p$2TZjNnOYNNQKpwLJqoGJbVsZG5/Fti8ovBRd.VzRbiDSl7TEq\
IaSMH.TeBKnTS/SjlMruW8qffC0JNORW.BTW1:16338:0:99999:7:::
tania:$6$8Z/zovxj$9qvoqT8i9KIrmN.k4EQwAF5ryz5yzNwEvYjAa9L5XVXQu.z4DlpvMREH\
eQpQzvRnqFdKkVj17H5ST.c79HDZw0:16356:0:99999:7:::
laura:$6$glDuTY5e$/NYYWLxfHgZFWeoujaXSMcR.Mz.lGOxtcxFocFVJNb98nbTPhWFXfKWG\
SyYh1WCv6763Wq54.w24Yr3uAZBOm/:16356:0:99999:7:::
valentina:$6$jrZa6PVI$1uQgqR6En9mZB6mKJ3LXRB4CnFko6LRhbh.v4iqUk9MVreui1lv7\
GxHOUDSKA0N55ZRNhGHa6T2ouFnVno/0o1:16356:0:99999:7:::
[root@centos7 ~]#
```

The **/etc/shadow** file contains nine colon separated columns. The nine fields contain (from left to right) the user name, the encrypted password (note that only inge and laura have an encrypted password), the day the password was last changed (day 1 is January 1, 1970), number of days the password must be left unchanged, password expiry day, warning number of days before password expiry, number of days after expiry before disabling the account, and the day the account was disabled (again, since 1970). The last field has no meaning yet.

All the passwords in the screenshot above are hashes of **hunter2**.

Encryption with passwd

Passwords are stored in an encrypted format. This encryption is done by the **crypt** function. The easiest (and recommended) way to add a user with a password to the system is to add the user with the **useradd -m user** command, and then set the user's password with **passwd**.

```
[root@RHEL4 ~]# useradd -m xavier
[root@RHEL4 ~]# passwd xavier Changing password
for user xavier.
New UNIX password:  Retype
new UNIX password:
passwd: all authentication tokens updated successfully.
[root@RHEL4 ~]#
```

encryption with openssl

Another way to create users with a password is to use the -p option of useradd, but that option requires an encrypted password. You can generate this encrypted password with the **openssl passwd** command.

The **openssl passwd** command will generate several distinct hashes for the same password, for this it uses a **salt**.

```
user@rhel65:~$ openssl passwd hunter2
86jcUNlnGDFpY user@rhel65:~$ openssl
passwd hunter2 Yj7mDO9OAnvq6
user@rhel65:~$ openssl passwd
hunter2
YqDcJeGoDbzKA
user@rhel65:~$
```

This **salt** can be chosen and is visible as the first two characters of the hash.

```
user@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
user@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
user@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
user@rhel65:~$
```

This example shows how to create a user with password.

```
root@rhel65:~# useradd -m -p $(openssl passwd hunter2) mohamed
```

groups

Users can be listed in **groups**. Groups allow you to set permissions on the group level instead of having to set permissions for every individual user.

Every Unix or Linux distribution will have a graphical tool to manage groups. Novice users are advised to use this graphical tool. More experienced users can use command line tools to manage users, but be careful: Some distributions do not allow the mixed use of GUI and CLI tools to manage groups (YaST in Novell Suse). Senior administrators can edit the relevant files directly with **vi** or **vigr**.

groupadd

Groups can be created with the **groupadd** command. The example below shows the creation of five (empty) groups.

```
root@predator-pc~#      groupadd      tennis
root@predator-pc~#     groupadd     football
root@predator-pc~#      groupadd      snooker
root@predator-pc~#     groupadd     formula1
root@predator-pc~# groupadd salsa
```

group file

Users can be a member of several groups. Group membership is defined by the **/etc/group** file.

```
root@predator-pc~#      tail   -5 /etc/group
    tennis:x:1006: football:x:1007:
snooker:x:1008: formula1:x:1009:
    salsa:x:1010: root@predator-pc~#
```

The first field is the group's name. The second field is the group's (encrypted) password (can be empty). The third field is the group identification or **GID**. The fourth field is the list of members, these groups have no members.

groups

A user can type the **groups** command to see a list of groups where the user belongs to.

```
[harry@RHEL4b ~]$ groups harry sports
[harry@RHEL4b ~]$
```

usermod

Group membership can be modified with the useradd or **usermod** command.

```
root@predator-pc~#  usermod  -a  -G  tennis  inge
root@predator-pc~#  usermod  -a  -G  tennis  katrien
root@predator-pc~#  usermod  -a  -G  salsa  katrien
root@predator-pc~#  usermod  -a  -G  snooker  sandra
root@predator-pc~# usermod -a -G formula1 annelies
root@predator-pc~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
snooker:x:1008:sandra
formula1:x:1009:annelies
salsa:x:1010:katrien root@predator-
pc~#
```

Be careful when using **usermod** to add users to groups. By default, the **usermod** command will **remove** the user from every group of which he is a member if the group is not listed in the command! Using the **-a** (append) switch prevents this behaviour.

groupmod

You can change the group name with the **groupmod** command.

```
root@predator-pc~# groupmod -n darts snooker
root@predator-pc~# tail -5 /etc/group
tennis:x:1006:inge,katrien football:x:1007:
formula1:x:1009:annelies salsa:x:1010:katrien
darts:x:1008:sandra
```

gpasswd

You can delegate control of group membership to another user with the **gpasswd** command. In the example below we delegate permissions to add and remove group members to serena for the sports group. Then we **su** to serena and add harry to the sports group.

```
[root@RHEL4b ~]# gpasswd -A serena sports

[root@RHEL4b ~]# su - serena [serena@RHEL4b ~]$
id harry uid=516(harry) gid=520(harry)
groups=520(harry)
[serena@RHEL4b ~]$ gpasswd -a harry sports
Adding user harry to group sports [serena@RHEL4b ~]$ id
harry uid=516(harry) gid=520(harry)
groups=520(harry),522(sports)
[serena@RHEL4b ~]$ tail -1 /etc/group sports:x:522:serena,venus,harry
[serena@RHEL4b ~]$
```

Group administrators do not have to be a member of the group. They can remove themselves from a group, but this does not influence their ability to add or remove members.

```
[serena@RHEL4b ~]$ gpasswd -d serena sports
Removing user serena from group sports [serena@RHEL4b ~]$ exit
```

Information about group administrators is kept in the **/etc/gshadow** file.

```
[root@RHEL4b ~]# tail -1 /etc/gshadow
sports:!:serena:venus,harry [root@RHEL4b
~]#
```

To remove all group administrators from a group, use the **gpasswd** command to set an empty administrators list.

```
[root@RHEL4b ~]# gpasswd -A "" sports
```

newgrp

You can start a **child shell** with a new temporary **primary group** using the **newgrp** command.

```
root@rhel65:~#        mkdir        prigroup

root@rhel65:~# cd prigroup/
root@rhel65:~/prigroup# touch standard.txt
root@rhel65:~/prigroup# ls -l total 0
-rw-r--r--. 1 root root 0 Apr 13 17:49 standard.txt root@rhel65:~/prigroup#
echo $SHLVL
1
root@rhel65:~/prigroup#             newgrp             tennis

root@rhel65:~/prigroup# echo $SHLVL
2
root@rhel65:~/prigroup# touch newgrp.txt
root@rhel65:~/prigroup# ls -l total 0
-rw-r--r--. 1 root tennis 0 Apr 13 17:49 newgrp.txt -
rw-r--r--. 1 root root   0 Apr 13 17:49 standard.txt
root@rhel65:~/prigroup# exit exit
root@rhel65:~/prigroup#
```

user profiles

Logged on users have a number of preset (and customized) aliases, variables, and functions, but where do they come from ? The **shell** uses a number of startup files that are executed (or rather **sourced**) whenever the shell is invoked. What follows is an overview of startup scripts.

system profile

Both the **bash** and the **ksh** shell will verify the existence of **/etc/profile** and **source** it if it exists.

When reading this script, you will notice (both on Debian and on Red Hat Enterprise Linux) that it builds the PATH environment variable (among others). The script might also change the PS1 variable, set the HOSTNAME and execute even more scripts like **/etc/inputrc** This screenshot uses grep to show PATH manipulation in **/etc/profile** on Debian.

```
root@debian7:~# grep PATH /etc/profile

PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin"
```

```
PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/game
s" export  PATH root@debian7:~#
```

This screenshot uses grep to show PATH manipulation in **/etc/profile** on RHEL7/CentOS7.

```
[root@centos7 ~]# grep PATH /etc/profile
case ":${PATH}:" in
                PATH=$PATH:$1
PATH=$1:$PATH export PATH USER LOGNAME MAIL HOSTNAME
HISTSIZE HISTCONTROL [root@centos7 ~]#
```

The **root user** can use this script to set aliases, functions, and variables for every user on the system.

bash_profile

When this file exists in the home directory, then **bash** will source it. On Debian Linux 5/6/7 this file does not exist by default.

RHEL7/CentOS7 uses a small ~/.**bash_profile** where it checks for the existence of ~/.**bashrc** and then sources it. It also adds $HOME/bin to the $PATH variable.

```
[root@rhel7 ~]# cat /home/user/.bash_profile
# .bash_profile

# Get the aliases and functions if
[ -f ~/.bashrc ]; then
        . ~/.bashrc fi # User specific environment

      and   startup     programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
[root@rhel7 ~]#
```

bash_login

When .**bash_profile** does not exist, then **bash** will check for ~/.**bash_login** and source it.

Neither Debian nor Red Hat have this file by default.

profile

When neither ~/.**bash_profile** and ~/.**bash_login** exist, then bash will verify the existence of ~/.**profile** and execute it. This file does not exist by default on Red Hat. On Debian this script can execute ~/.**bashrc** and will add $HOME/bin to the $PATH variable.

```
root@debian7:~# tail -11 /home/user/.profile
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists

if [ -f "$HOME/.bashrc" ]; then
. "$HOME/.bashrc"     fi fi

# set PATH so it includes user's private bin if it exists
if     [     -d     "$HOME/bin"     ]     ;     then

PATH="$HOME/bin:$PATH" fi
```

RHEL/CentOS does not have this file by default.

bashrc

The ~/.**bashrc** script is often sourced by other scripts. Let us take a look at what it does by default.

Red Hat uses a very simple ~/.**bashrc**, checking for /**etc**/**bashrc** and sourcing it. It also leaves room for custom aliases and functions.

```
[root@rhel7 ~]# cat /home/user/.bashrc

# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc

fi
# Uncomment the following line if you don't like systemctl's auto-paging feature:

# export SYSTEMD_PAGER=

# User specific aliases and functions
```

On Debian this script is quite a bit longer and configures $PS1, some history variables and a number af active and inactive aliases.

```
root@debian7:~# wc -l /home/user/.bashrc
110 /home/user/.bashrc
```

bash_logout

When exiting **bash**, it can execute ~/.**bash_logout**.

Debian use this opportunity to clear the console screen.

```
serena@deb503:~$ cat .bash_logout # ~/.bash_logout: executed by bash(1) when login
shell exits. # when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q fi
```

Red Hat Enterprise Linux 5 will simple call the /**usr**/**bin**/**clear** command in this script.

```
[serena@rhel53 ~]$ cat .bash_logout
# ~/.bash_logout /usr/bin/clear
```

Red Hat Enterprise Linux 6 and 7 create this file, but leave it empty (except for a comment).

```
user@rhel65:~$ cat .bash_logout
# ~/.bash_logout
```

CHAPTER 7

Desktop review

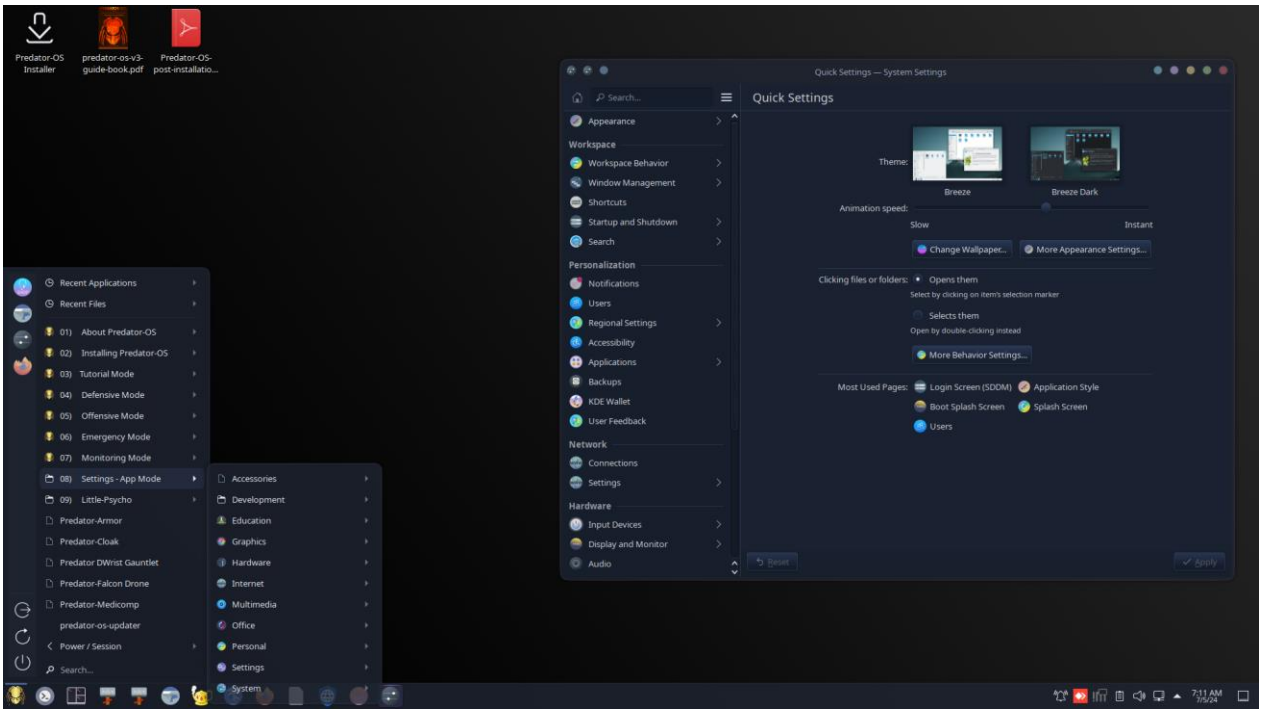
Chapter 7

Desktop review

KDE Plasma Desktop, also referred to simply as Plasma, is a graphical user interface (GUI) environment designed primarily for Linux systems [Wikipedia KDE Plasma 5]. It's known for being customizable and user-friendly.

Here are some key features of Plasma Desktop:

- **Highly Customizable:** Plasma allows you to arrange the desktop layout exactly how you like it, with panels, widgets (also called plasmoids), and virtual desktops [KDE Plasma desktop].
- **Powerful Search:** The built-in search function, KRunner, lets you quickly launch applications, find files, perform calculations and conversions, and more [KDE Plasma desktop].
- **Easy on Resources:** Plasma is known for being lightweight and efficient, so it can run well on older computers [KDE Plasma desktop].



About Predator-OS

See the first item in menu that is for introducing of Predator-OS.

Predator-OS Menu hierarchy

- Applications
- Accessibility
- Accessories
- Development
- Education
- Games
- Graphics
- Internet
- Multimedia
- Office
- Other
- Science
- System

Theming

Plasma themes can be stored in two locations on your system:

- **System/Default Themes**: These are themes that come pre-installed with your operating system and are accessible to all users. You'll find them in this directory:

  /usr/share/plasma/desktoptheme/

- **User Installed Themes**: These are themes that you've downloaded and installed yourself. They are stored in your user profile directory:

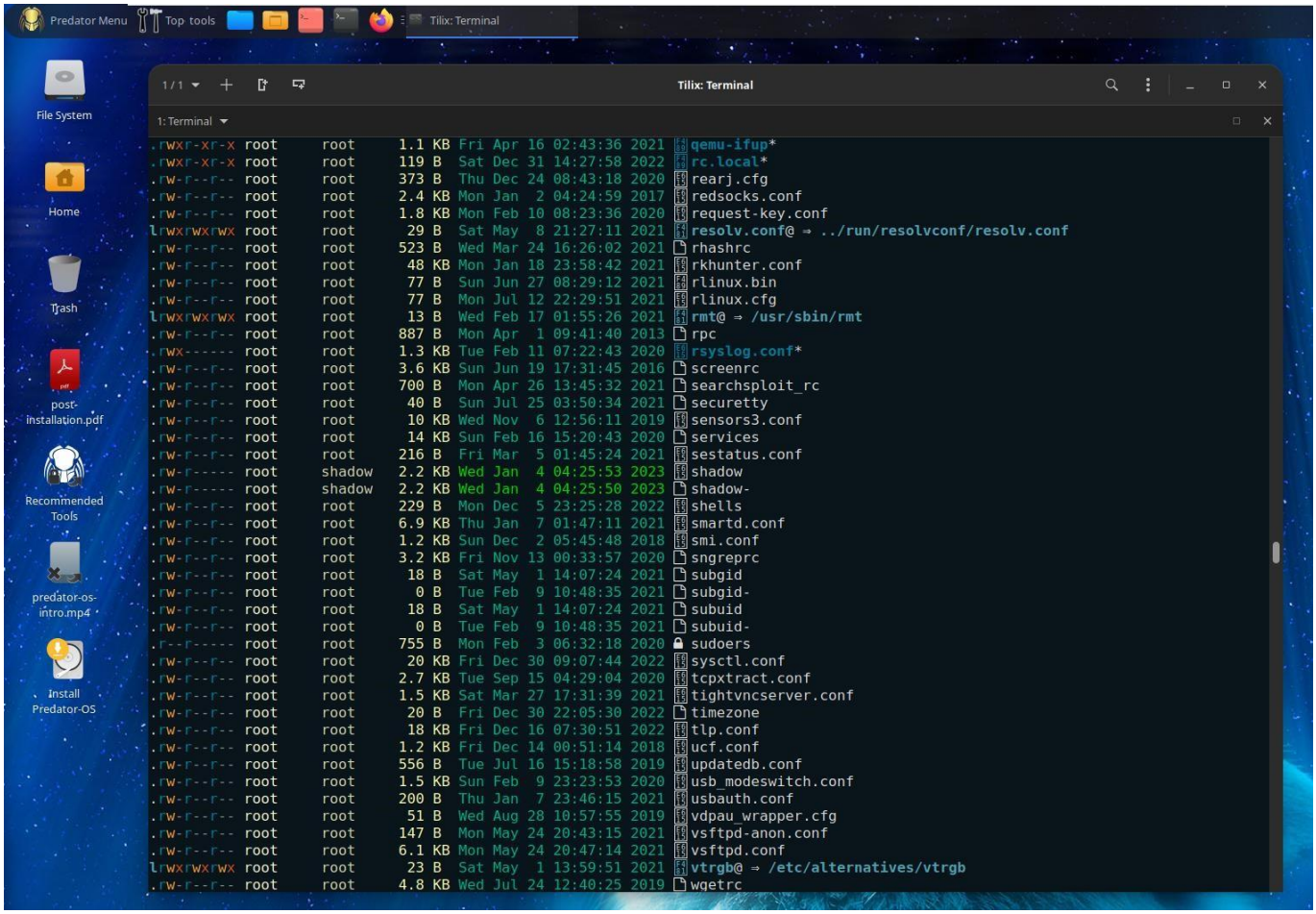
  ~/.local/share/plasma/desktoptheme/

File types

Most filesystem implementations define seven types of files. Even when developers add something new and wonderful to the file tree (such as the process information under /**proc**), it must still be made to look like one of these seven types:

- Regular files
- Directories
- Character device files
- Block device files
- Local domain sockets
- Named pipes (FIFOs)
- Symbolic links
- File-type encoding used by ls

File type	Symbol	Created by	Removed by
Regular file	-	editors, **cp**, etc.	rm
Directory	d	mkdir	rmdir, rm -r
Character device file	c	mknod	rm
Block device file	b	mknod	rm
Local domain socket	s	**socket** system call	rm
Named pipe	p	mknod	rm
Symbolic link	l	ln -s	rm



Pathname	Contents
/bin	Core operating system commands
/boot	Boot loader, kernel, and files needed by the kernel
/compat	On FreeBSD, files and libraries for Linux binary compatibility
/dev	Device entries for disks, printers, pseudo-terminals, etc.
/etc	Critical startup and configuration files
/home	Default home directories for users
/lib	Libraries, shared libraries, and commands used by /bin and /sbin
/media	Mount points for filesystems on removable media
/mnt	Temporary mount points, mounts for removable media
/opt	Optional software packages (rarely used, for compatibility)
/proc	Information about all running processes
/root	Home directory of the superuser (sometimes just /)
/run	Rendezvous points for running programs (PIDs, sockets, etc.)

/sbin	Core operating system commands a
/srv	Files held for distribution through web or other servers
/sys	A plethora of different kernel interfaces (Linux)
/tmp	Temporary files that may disappear between reboots
/usr	Hierarchy of secondary files and commands
/usr/bin	Most commands and executable files
/usr/include	Header files for compiling C programs
/usr/lib	Libraries; also, support files for standard programs
/usr/local	Local software or configuration data; mirrors /usr
/usr/sbin	Less essential commands for administration and repair
/usr/share	Items that might be common to multiple systems
usr/share/man	On-line manual pages
/usr/src	Source code for nonlocal software (not widely used)
/usr/tmp	More temporary space (preserved between reboots)
/var	System-specific data and a few configuration files
/var/adm	Varies: logs, setup records, strange administrative bits
/var/log	System log files
/var/run	Same function as /run; now often a symlink
/var/spool	Spooling (that is, storage) directories for printers, mail, etc.
/var/tmp	More temporary space (preserved between reboots)

PREDATOR-OS

SECURITY, PRIVACY, HARDENED ANONYMITY

CHAPTER 8

Post installation

Chapter 8

Post installation

After the First Boot

Maintenance and Updates: The APT Tools

APT is the abbreviation for Advanced Packaging Tool. What makes this program "advanced" is its approach to packages. It does not simply evaluate them individually, but it considers them as a whole and produces the best possible combination of packages depending on what is available and compatible according to dependencies.

You can also update Predator-OS using the terminal.

```
Press CTRL+ALT+T to bring up a Terminal window.
```

Then type the followings command:

```
$sudo apt update
```

Or

```
$up
```

Mange the sources.list

A first apt upgrade (a command used to automatically update installed programs) is generally required, especially for possible security updates issued since the release of the latest PredatorOS stable version.

Befor updating and upgrading the Predator-OS:

The Predator-OS used the full repository of Debian stable 20.04 by default.you can see the list of repositories in the following path:

```
/etc/apt/sources.list
```

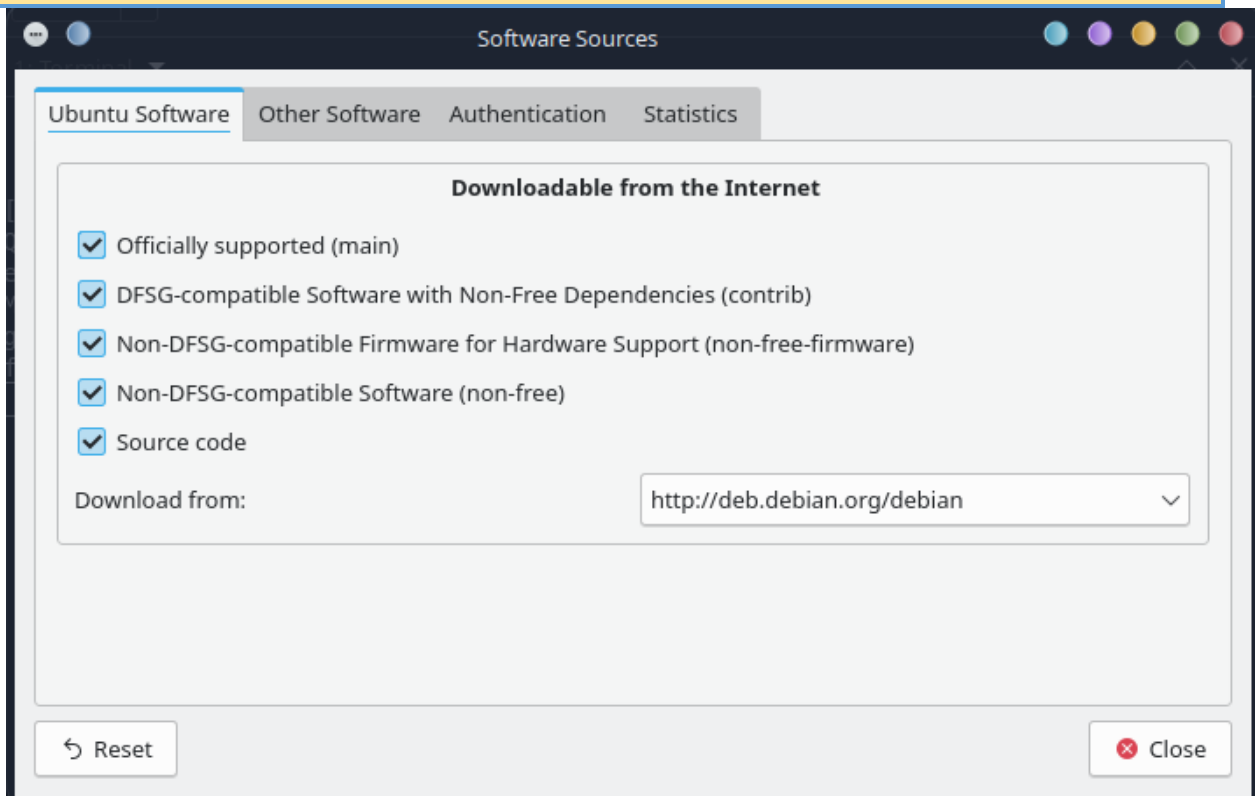
```
##Debian 12.5 bookworm version
deb http://deb.debian.org/debian/ bookworm non-free-firmware non-free contrib main
deb-src http://deb.debian.org/debian/ bookworm non-free-firmware non-free contrib main

deb http://deb.debian.org/debian-security/ bookworm-security non-free-firmware non-free contrib main
deb-src http://deb.debian.org/debian-security/ bookworm-security non-free-firmware non-free contrib main

deb http://deb.debian.org/debian/ bookworm-updates non-free-firmware non-free contrib main
deb-src http://deb.debian.org/debian/ bookworm-updates main contrib non-free non-free-firmware

deb https://deb.debian.org/debian/ bookworm-proposed-updates contrib main non-free non-free-firmware
deb-src https://deb.debian.org/debian/ bookworm-proposed-updates contrib main non-free non-free-firmware



#Debian testing repository
# deb http://deb.debian.org/debian/ bookworm-backports main contrib non-free non-free-firmware
# deb-src http://deb.debian.org/debian/ bookworm-backports main contrib non-free non-free-firmware


#Debian 13  trixie
# deb https://deb.debian.org/debian/ trixie main contrib
# deb http://mirrors.kernel.org/debian/ trixie contrib main
```

Each active line in the **/etc/apt/sources.list** file represents a package source (repository) and is made of at least three parts separated by spaces.

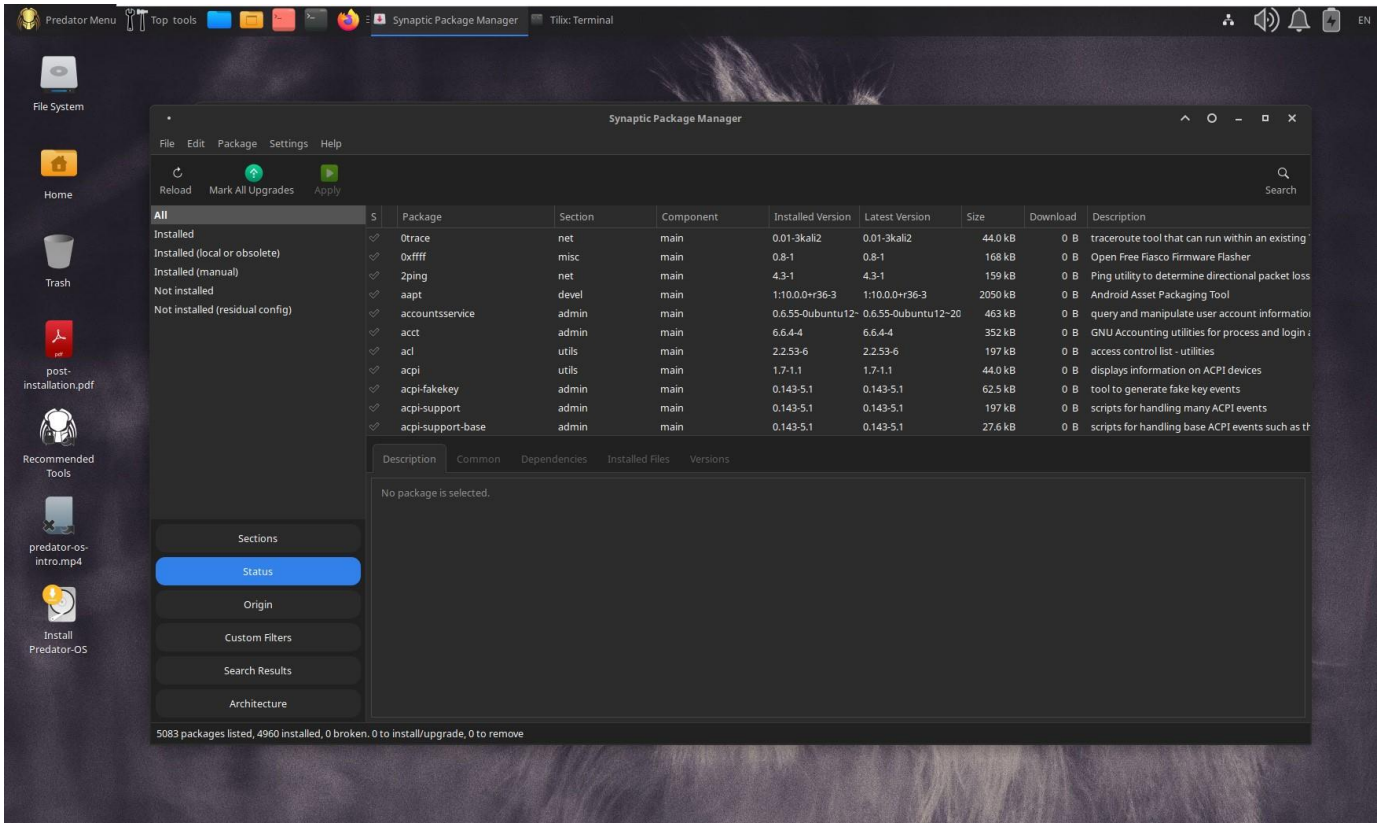
You can manage the repositories by adding or removing lines. sources.list management by GUI tools:

method1 :you can manage the repositories by typing the following command:

```
$sudo software-properties-qt
```



method2: you can manage the repositories by Synaptic tool.



Debian stable sources list in Predator-OS

```
##Debian 12.5 bookworm version
deb http://deb.debian.org/debian/ bookworm non-free-firmware non-free contrib main
deb-src http://deb.debian.org/debian/ bookworm non-free-firmware non-free contrib main

deb http://deb.debian.org/debian-security/ bookworm-security non-free-firmware non-free contrib main
deb-src http://deb.debian.org/debian-security/ bookworm-security non-free-firmware non-free contrib main

deb http://deb.debian.org/debian/ bookworm-updates non-free-firmware non-free contrib main
deb-src http://deb.debian.org/debian/ bookworm-updates main contrib non-free non-free-firmware

deb https://deb.debian.org/debian/ bookworm-proposed-updates contrib main non-free non-free-firmware
deb-src https://deb.debian.org/debian/ bookworm-proposed-updates contrib main non-free non-free-firmware
```

Own Predator-os source list

```
deb [arch=amd64] https://www.seilany.ir/predator-os/predator-updater-ppa ./
```

Upgrading the System

Apt or apt-get

APT is a vast project, whose original plans included a graphical interface. It is based on a library which contains the core application, and **apt-get** is the first front end — command-line based — which was developed within the project. **apt** is a second command-line based front end provided by APT which overcomes some design mistakes of **apt-get**.

Both tools are built on top of the same library and are thus very close, but the default behavior of **apt** has been improved for interactive use and to actually do what most users expect. The APT developers reserve the right to change the public interface of this tool to further improve it. Conversely, the public interface of **apt-get** is well defined and will not change in any backwards incompatible way. It is thus the tool that you want to use when you need to script package installation requests.

Numerous other graphical interfaces then appeared as external projects: **synaptic**, **aptitude** (which includes both a text mode interface and a graphical one — even if not complete yet), **wajig**, etc. The most recommended interface, **apt**, is the one that we will use in the examples given in this section. Note, however, that **apt-get** and **aptitude** have a very similar command line syntax. When there are major differences between these three commands, these will be detailed.

Type in:

```
sudo apt update
```

or use pre-definded alias command:

```
$up
```

You will be prompted to enter your login password.

System Upgrade

Regular upgrades are recommended, because they include the latest security updates. To upgrade, use **apt upgrade**, **apt-get upgrade** or **aptitude safeupgrade** (of course after **apt update**). This command looks for installed packages which can be upgraded without removing any packages. In other words, the goal is to ensure the least intrusive upgrade possible. **apt-get** is slightly more demanding than **aptitude** or **apt** because it will refuse to install packages which were not installed beforehand.

apt will generally select the most recent version number (except for packages from Experimental and stable-backports, which are ignored by default whatever their version number).
To tell **apt** to use a specific distribution when searching for upgraded packages, you need to use the **-t** or **--target-release** option, followed by the name of the distribution you want. To avoid specifying this option every time you use **apt**. For more important upgrades, such as the change from one major version to the next, you need to use **apt full-upgrade**. With this instruction, **apt** will complete the upgrade even if it has to remove some obsolete packages or install new dependencies.
Unlike **apt** and **aptitude**, **apt-get** does not know the **full-upgrade** command. Instead, you should use **apt-get dist-upgrade** ("distribution upgrade"), the historical and well-known command that **apt** and **aptitude** also except for the convenience of users who got used to it. The results of these operations are logged into **/var/log/apt/history.log** and **/var/log/apt/term.log**, whereas **dpkg** keeps its log in a file called **/var/log/dpkg.log**.

This will check for updates and tell you if there are any that need applying. To apply any updates type:

```
sudo apt dist-upgrade
```

or use pre-definded alias command:

```
$ug
```

Unattended Upgrades unattended-upgrades

The purpose of unattended-upgrades is to keep the computer current with the latest security (and other) updates automatically.

If you plan to use it, you should have some means to monitor your systems such as installing the apt-listchanges package and configuring it to send you emails about updates. And there is always /var/log/dpkg.log, or the files in /var/log/unattendedupgrades/.

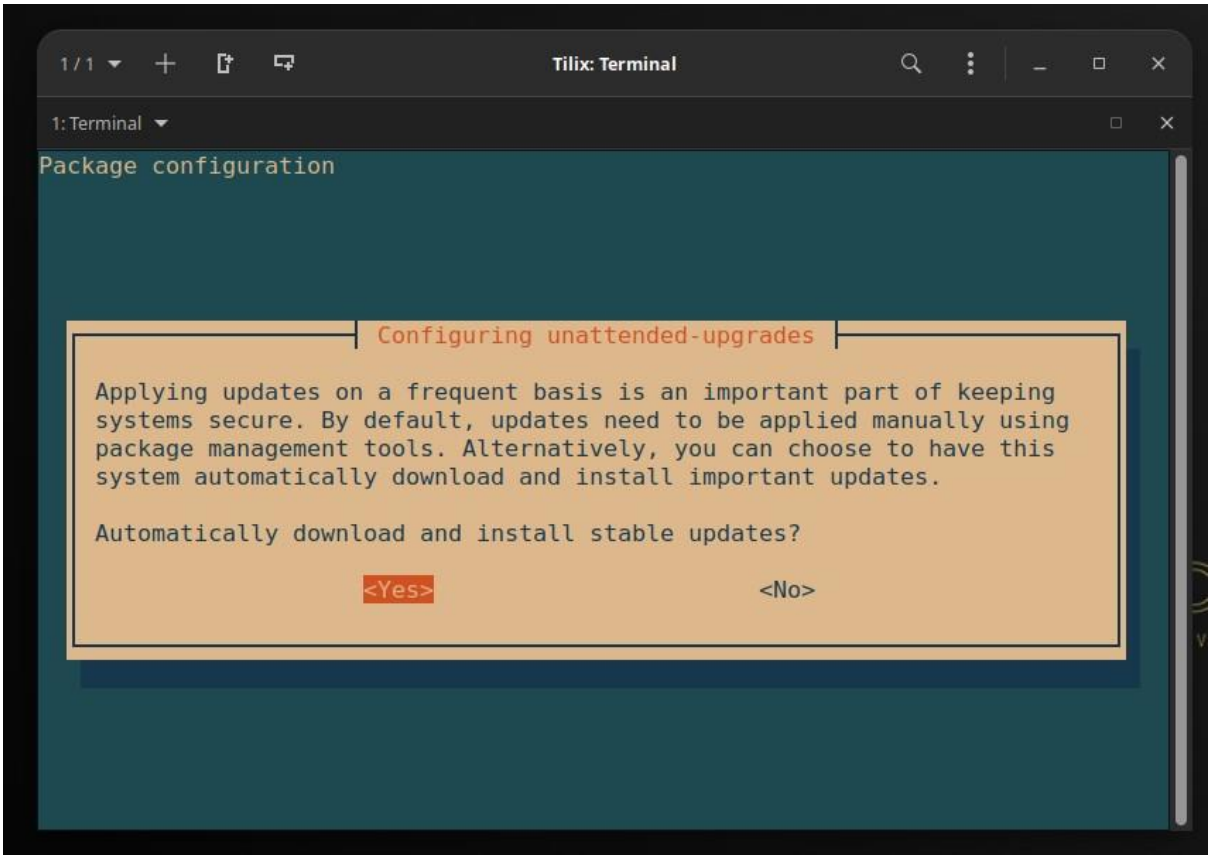
Linux server security is of critical importance to sysadmins. One central part of keeping Linux servers secure is by installing security updates promptly. Too often, there are compromised servers on the internet due to pending security updates waiting for a manual update. On both Debian stable and Debian, the unattended-upgrades package can be configured to perform unattended-upgrades to install updated packages and security updates automatically.

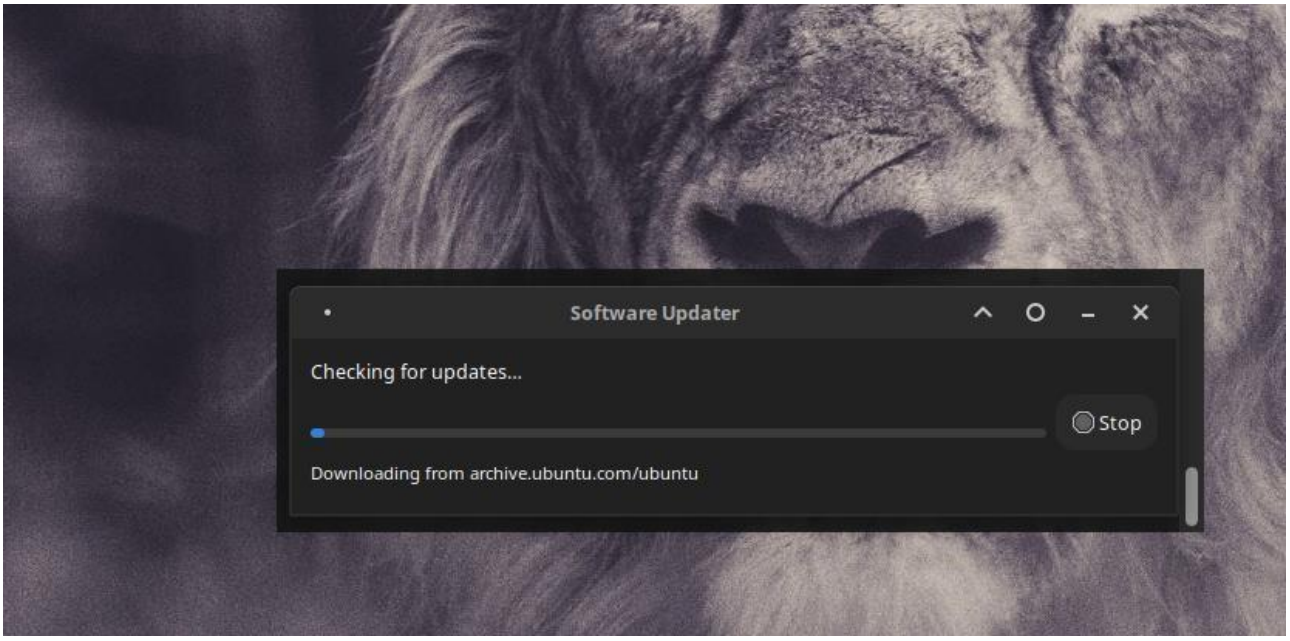
Remember, you will want to monitor updates and changes to your Linux server over time. You can monitor via **/var/log/dpkg.log** or read the log files in **/var/log/unattended-upgrades/**. You can also monitor changes by installing the apt-listchanges package (optional).

```
sudo apt install apt-listchanges
```

```
apt-get install unattended-upgrades apt-listchanges sudo dpkg-reconfigure unattended-upgrades he
```

apt-listchanges can be configured to send emails about update changes. apt-listchanges is a tool to show what has been changed in a new version of a Debian package compared to the version currently installed on the system. It does this by extracting the relevant entries from the NEWS.Debian and changelog[.Debian] files, usually found in /usr/share/doc/*package*, from Debian package archives. On both Debian and Debian stable, as Debian stable is a derivative of Debian.





The apt-cache Command

The **apt-cache** command can display much of the information stored in APT's internal database. This information is a sort of cache since it is gathered from the different sources listed in the **sources.list** file. This happens during the **apt update** operation.

Configuration Options

Besides the configuration elements already mentioned, it is possible to configure certain aspects of APT by adding directives in a file of
the **/etc/apt/apt.conf.d/** directory or **/etc/apt/apt.conf** itself

Installing Additional Software

The installed packages correspond to the profiles selected during installation, but not necessarily to the use that will actually be made of the machine. As such, you might want to use a package management tool to refine the selection of installed packages. The two most frequently used tools are apt (accessible from the command line) and synaptic ("Synaptic Package Manager" in the menus).

Manipulating Packages with dpkg:

dpkg is the base command for handling Debian packages on the system. If you have .deb packages, it is dpkg that allows installation or analysis of their contents. it knows what is installed on the system, and whatever it is given on the command line, but knows nothing of the other available packages. As such, it will fail if a

dependency is not met. Tools such as apt and aptitude, on the contrary, will create a list of dependencies to install everything as automatically as possible. Installing Packages dpkg is, above all, the tool for installing an already available Debian package (because it does not download anything). To do this, we use its -i or --install option.

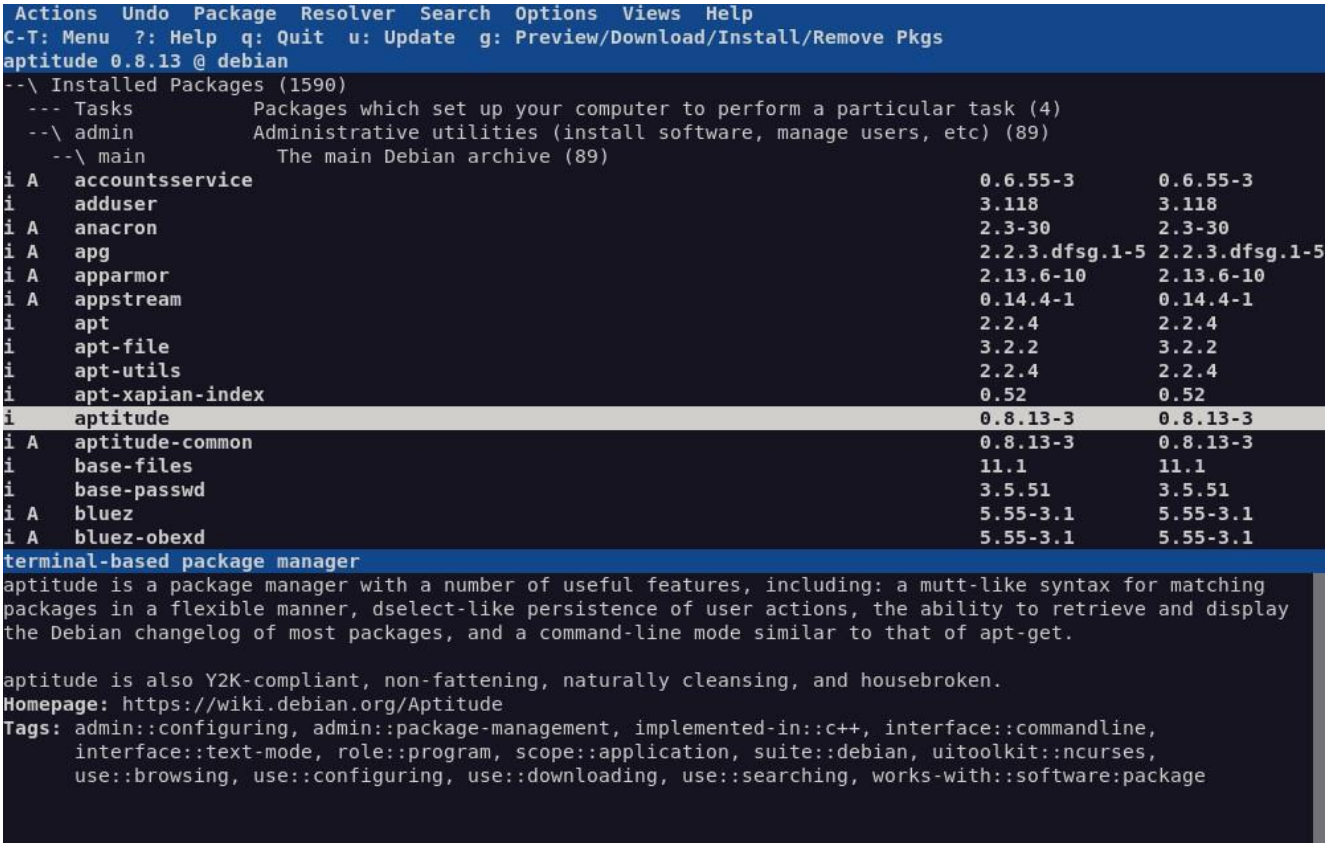
```
$sudo  dpkg -i myfile.deb
```

Frontends: aptitude, synaptic

APT is a C++ program whose code mainly resides in the **libapt-pkg** shared library. Using a shared library facilitates the creation of user interfaces (frontends), since the code contained in the library can easily be reused. Historically, **apt-get** was only designed as a test frontend for **libapt-pkg** but its success tends to obscure this fact.

aptitude

aptitude is an interactive program that can be used in semi-graphical mode on the console. You can browse the list of installed and available packages, look up all the available information, and select packages to install or remove. The program is designed specifically to be used by administrators, so that its default behaviors are designed to be much more intelligent than **apt-get**'s and its interface much easier to understand.

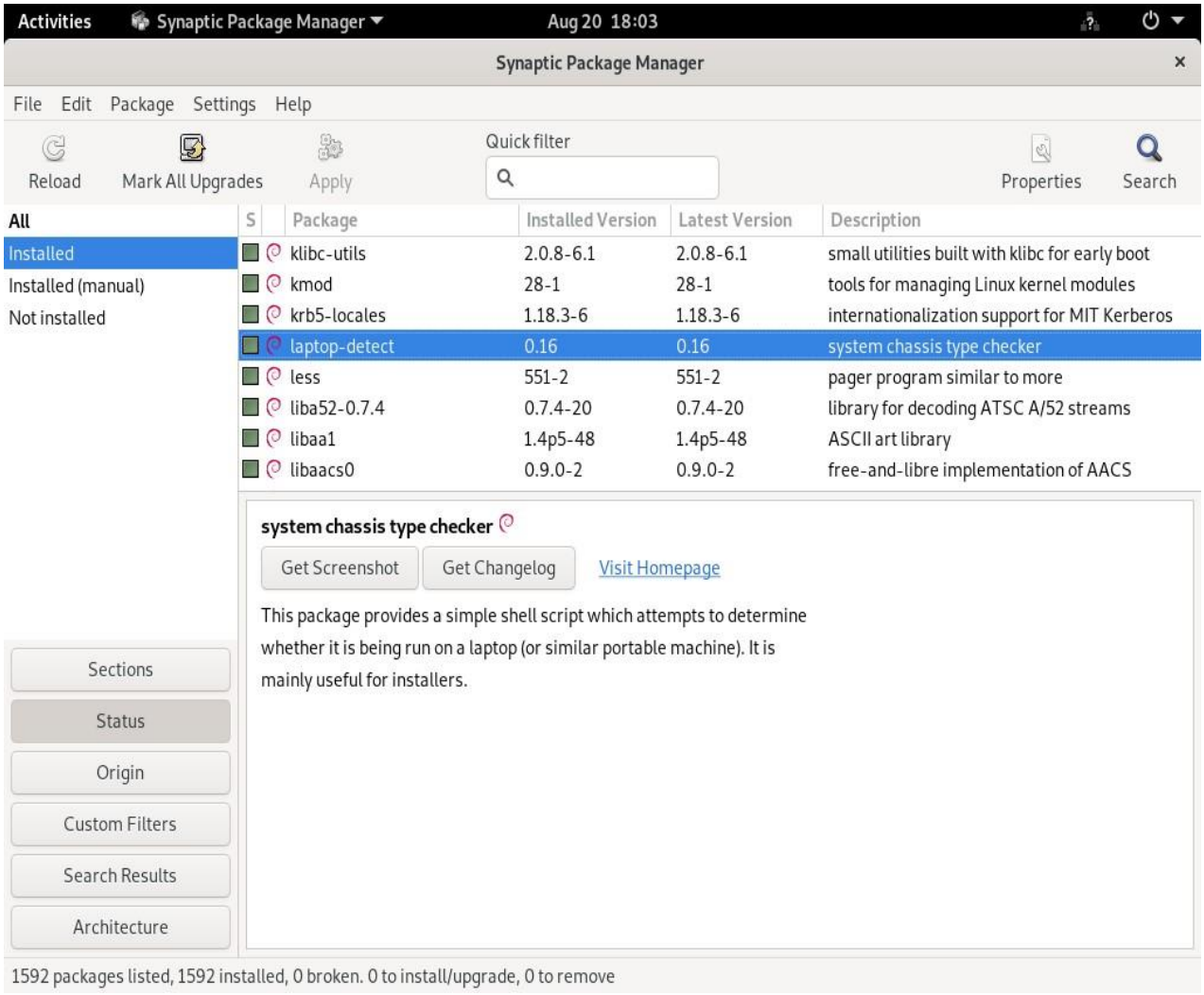


The aptitude package manager

When it starts, **aptitude** shows a list of packages sorted by state (installed, noninstalled, or installed but not available on the mirrors — other sections display tasks, virtual packages, and new packages that appeared recently on mirrors). To facilitate thematic browsing other views are available. In all cases, **aptitude** displays a list combining categories and packages on the screen. Categories are organized through a tree structure, whose branches can respectively be unfolded or closed with the **Enter**, **[** and **]** keys. + should be used to mark a package for installation, - to mark it for removal and _ to purge it (note that these keys can also be used for categories, in which case the corresponding actions will be applied to all the packages of the category). **u** updates the lists of available packages and **Shift**+**u** prepares a global system upgrade. **g** switches to a summary view of the requested changes (and typing **g** again will apply the changes), and **q** quits the current view. If you are in the initial view, this will effectively close **aptitude**.

Synaptic

synaptic is a graphical package manager which features a clean and efficient graphical interface based on GTK+/GNOME. Its many ready-to-use filters give fast access to newly available packages, installed packages, upgradable packages, obsolete packages and so on. If you browse through these lists, you can select the operations to be done on the packages (install, upgrade, remove, purge); these operations are not performed immediately, but put into a task list. A single click on a button then validates the operations, and they are performed in one go.



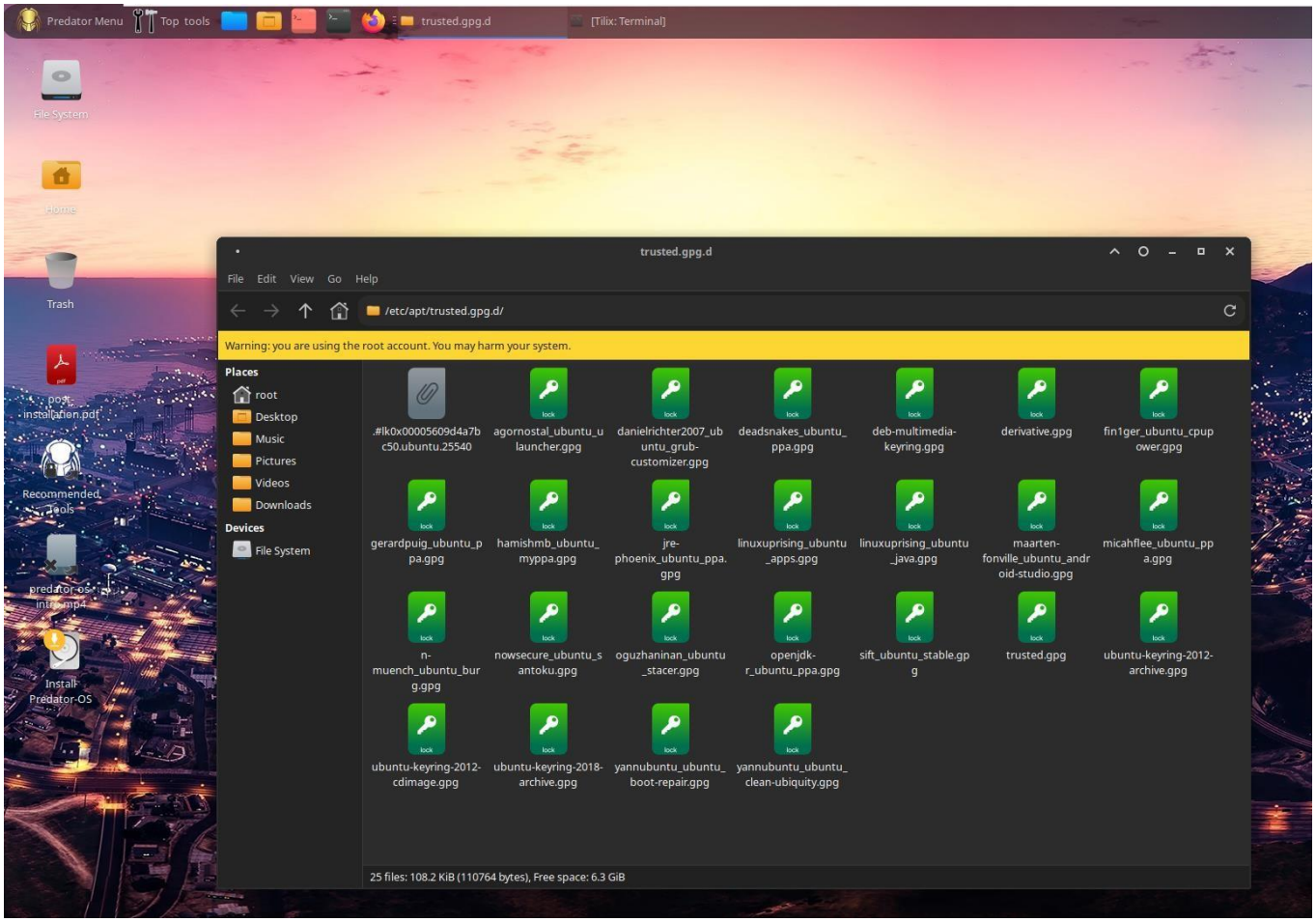
synaptic package manager

Checking Package Authenticity

Security is very important for administrators. Accordingly, they need to ensure that they only install packages that are guaranteed with no tampering on the way. A computer cracker could try to add malicious code to an otherwise legitimate package. Such a package, if installed, could do anything the cracker designed it to do, including for instance disclosing passwords or confidential information.

APT needs a set of trusted GnuPG public keys to verify signatures in the InRelease and Release.gpg files available on the mirrors. It gets them from files in /etc/apt/trusted.gpg.d/ and from the /etc/apt/trusted.gpg keyring (managed by the apt-key command).

```
# ls /etc/apt/trusted.gpg.d/
```



PREDATOR-OS

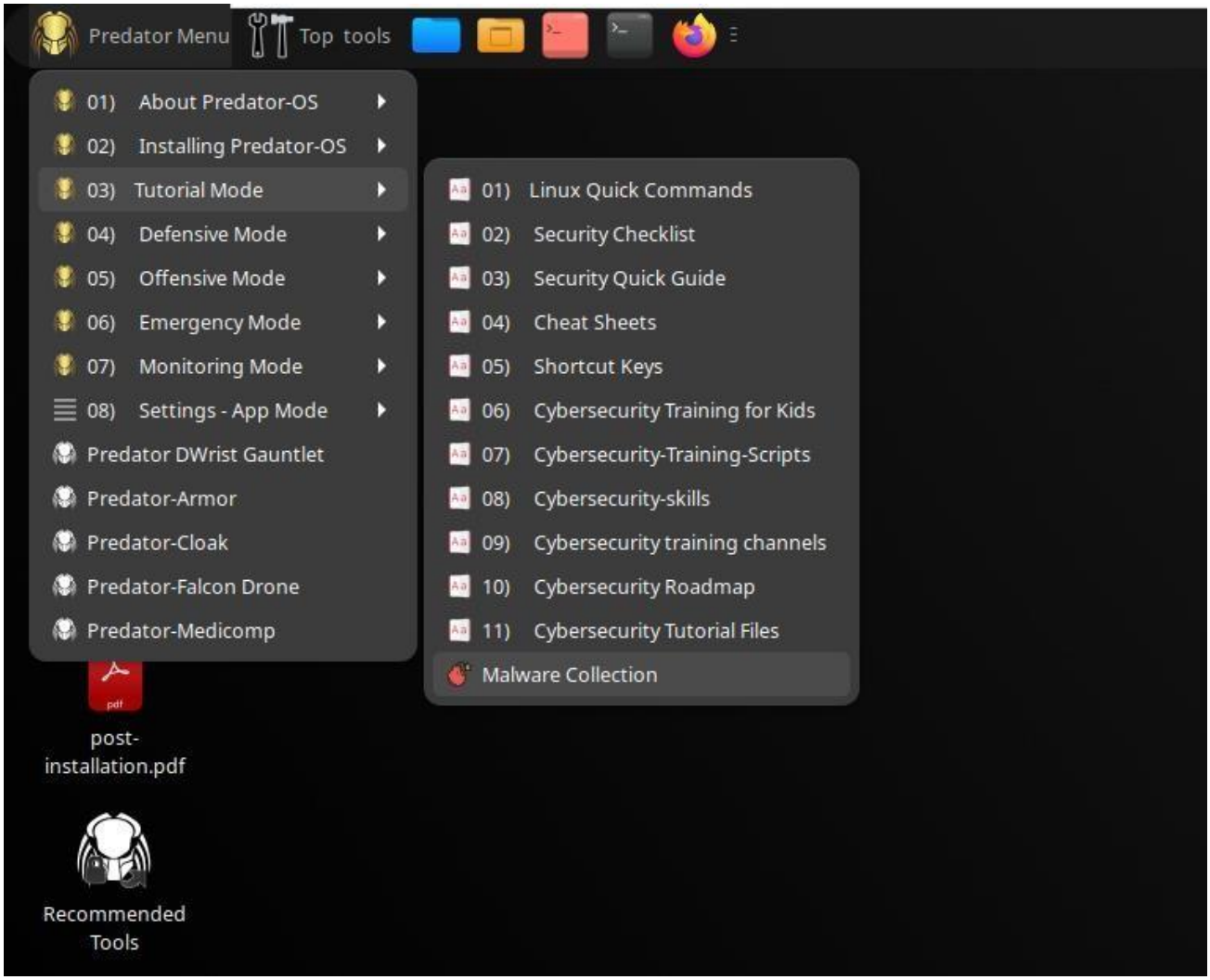
SECURITY, PRIVACY, HARDENED ANONYMITY

CHAPTER 9

Predator-OS modes

Chapter 9

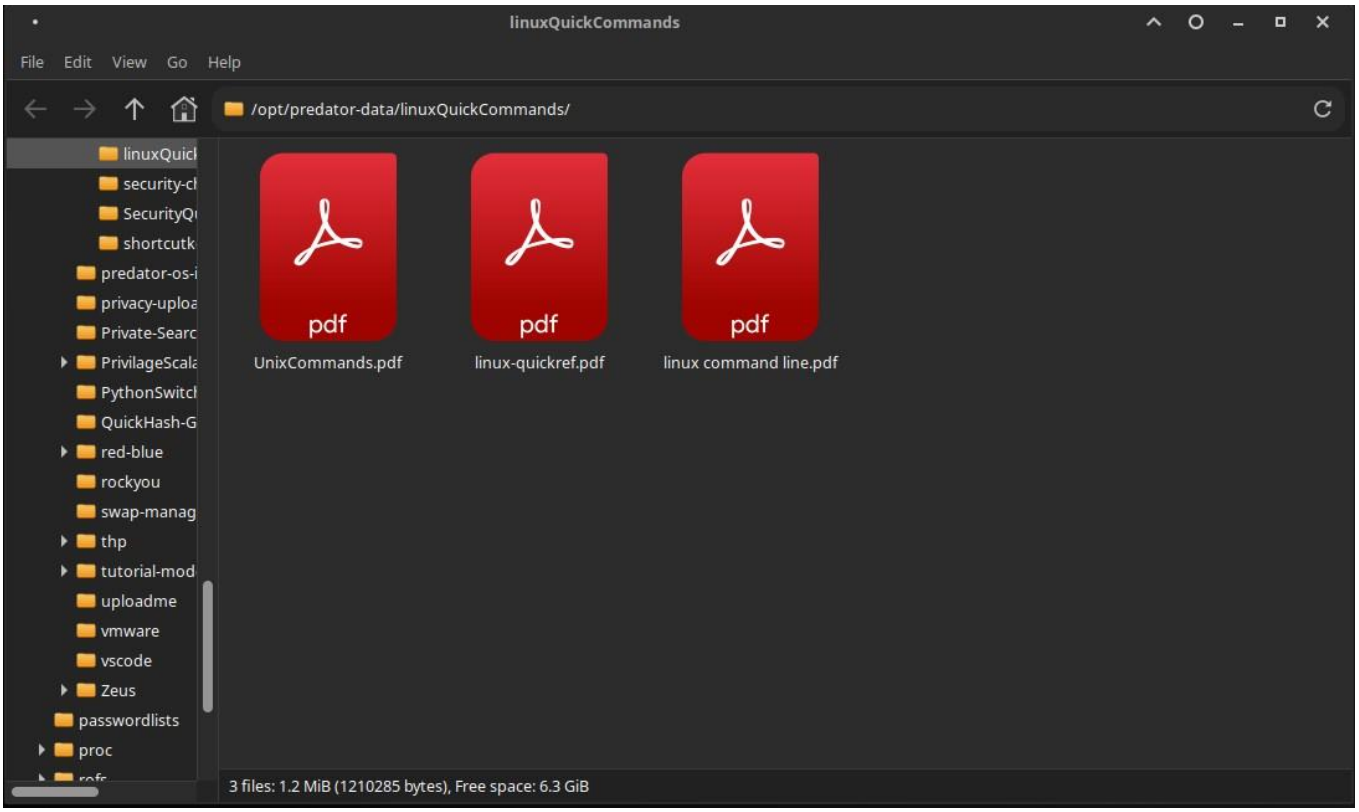
Predator-OS modes
Tutorial Mode



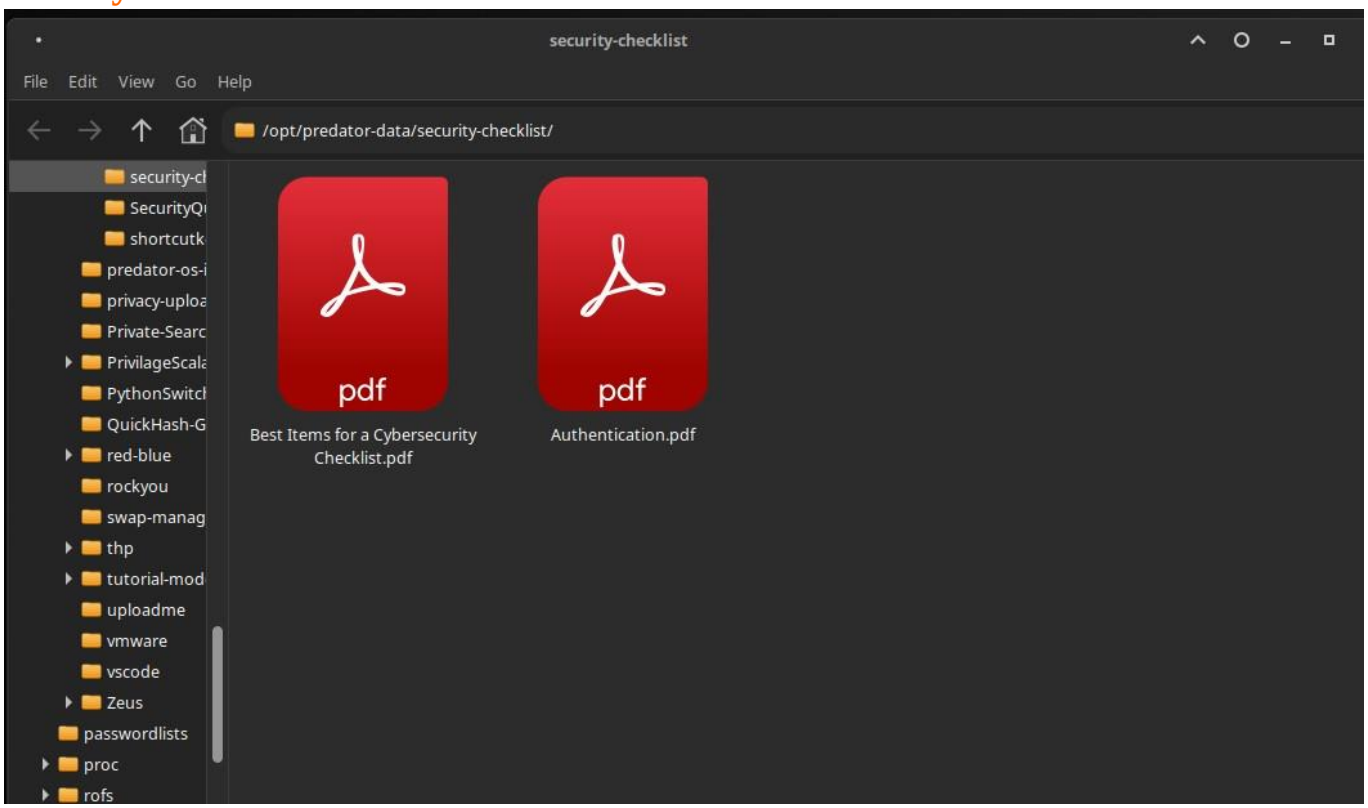
You have the following built-in educational files in the tutorial mode.

1)	Linux Quick Commands
2)	Security Check List
3)	Security Quick Guide
4)	Cheat Sheet
5)	Shortcut Keys
6)	Cybersecurity Training for Kids
7)	Cybersecurity Training Scripts
8)	Cybersecurity Skills
9)	Cybersecurity Training Channels
10)	Cybersecurity Roadmaps
11)	Cybersecurity Tutorial Files

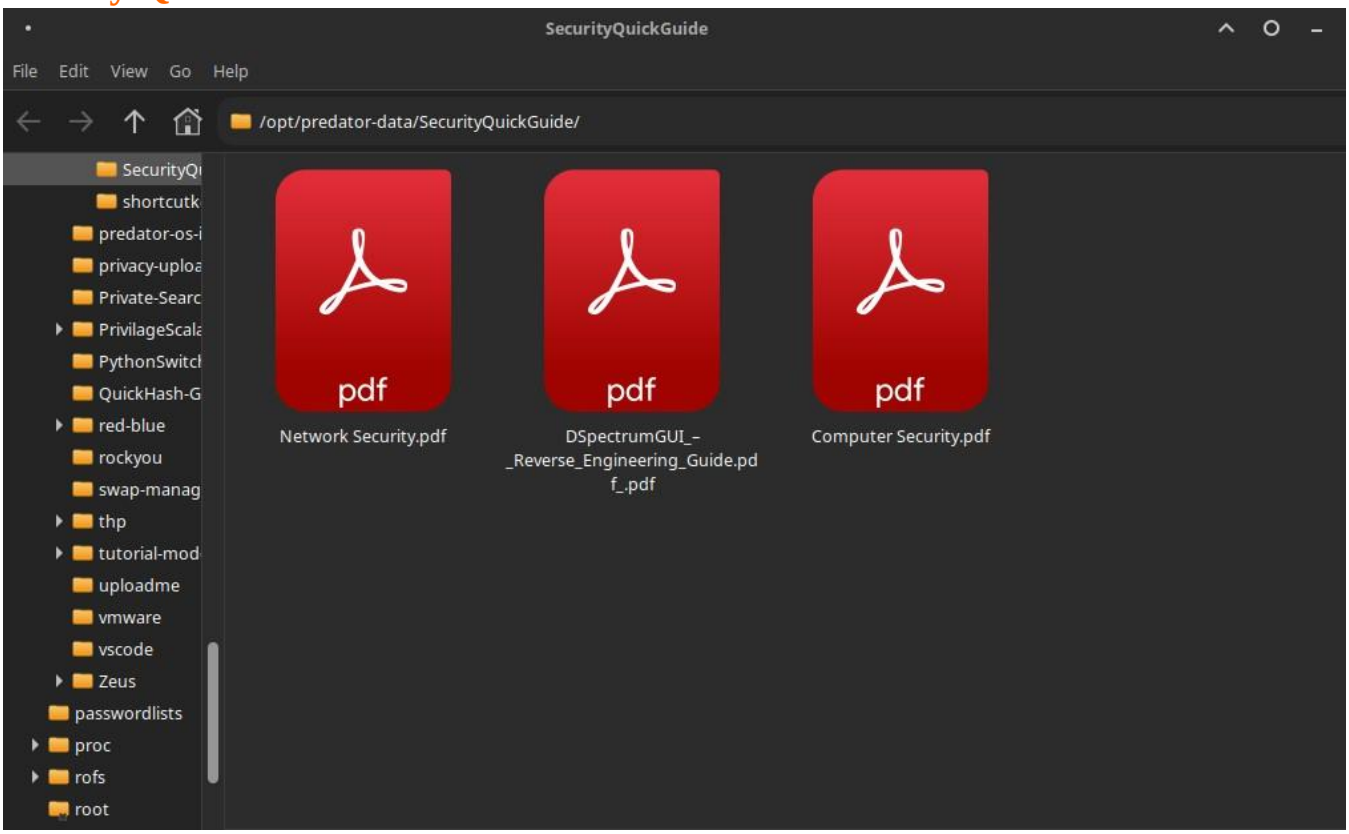
Linux Quick Commands



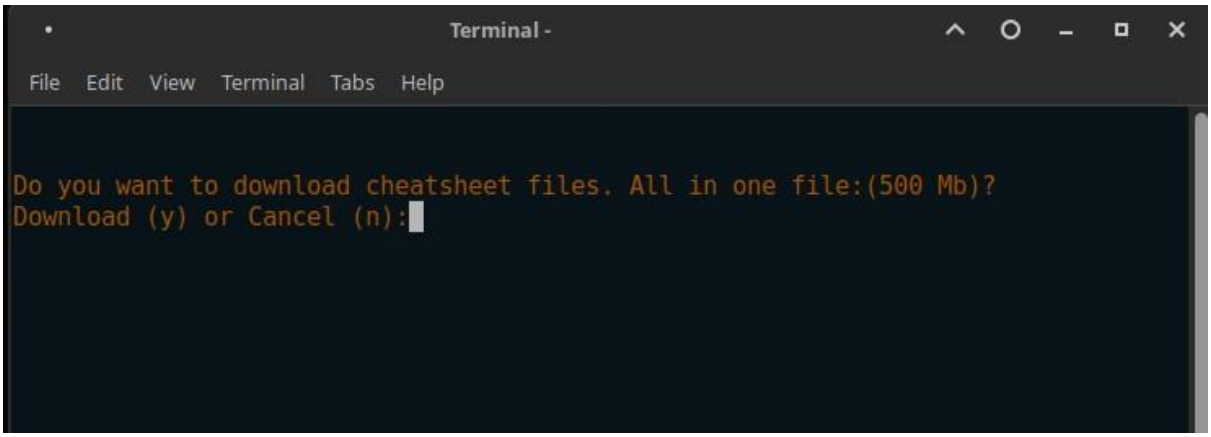
Security Check List



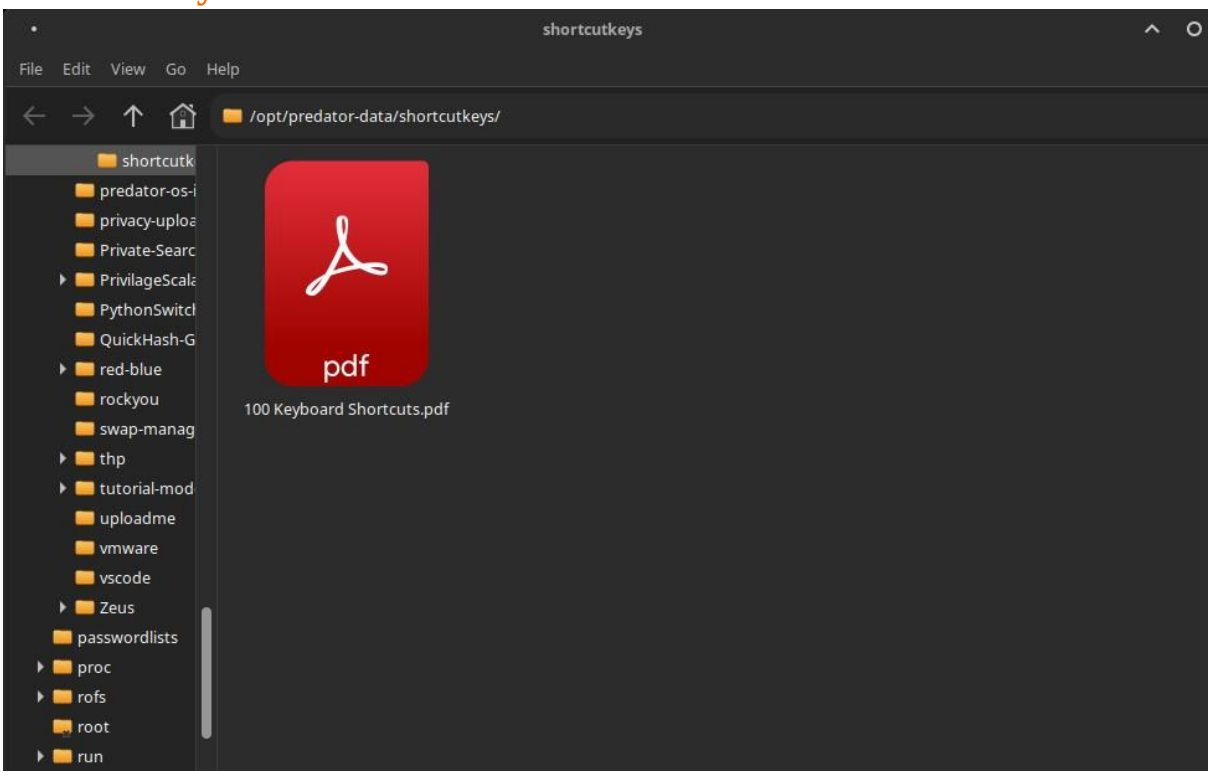
Security Quick Guide



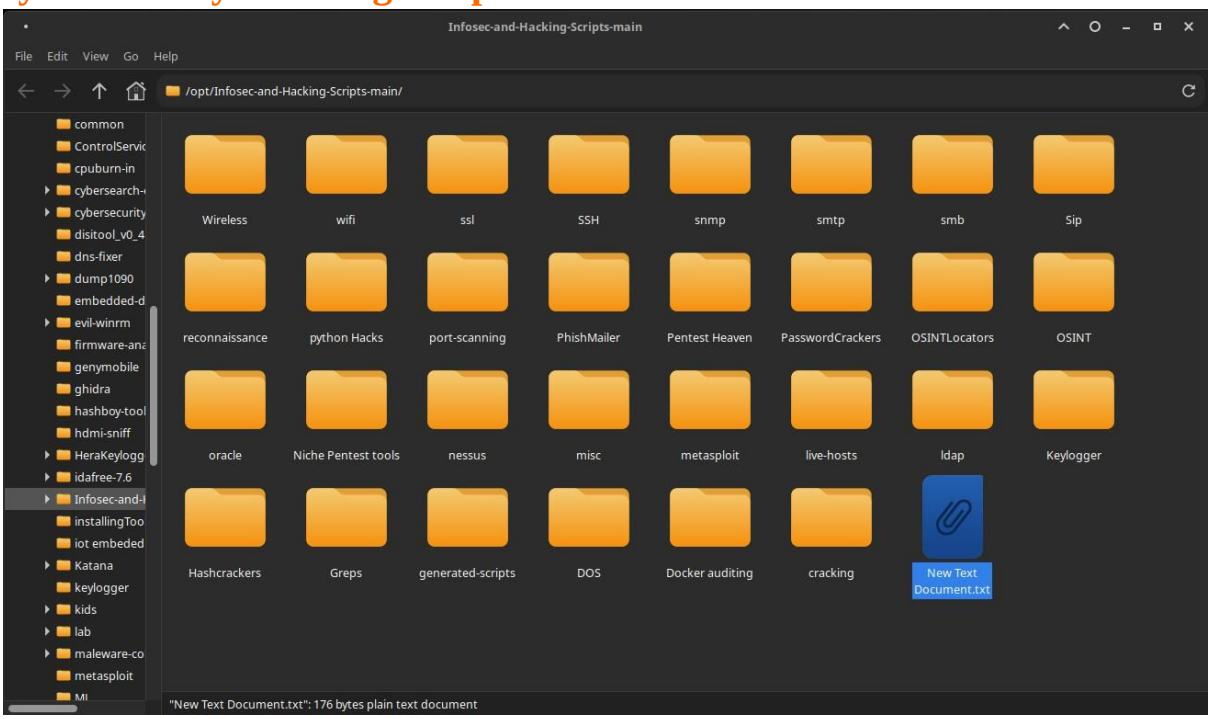
Cheat Sheet



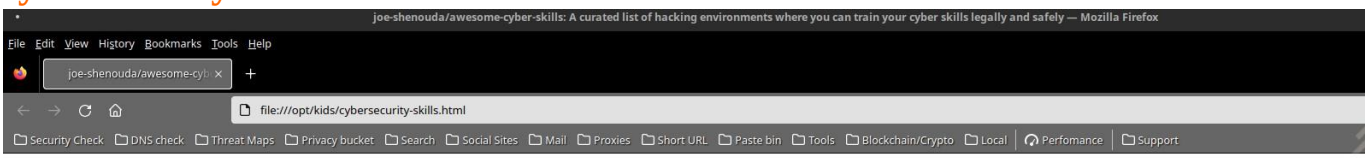
Shortcut Keys



Cybersecurity Training Scripts



Cybersecurity Skills



CYBERSECURITY-Kids:

For everyone in the Information Security business, it's important to understand the enemy, the hacker. Understanding the enemy makes you the best defender you can be to secure the digital world

By knowing your enemy, you can defeat your enemy.

Training your cyber skills means also keeping your hacking skills up to date. To do this, you need an environment to practice in, free, legally and safely.

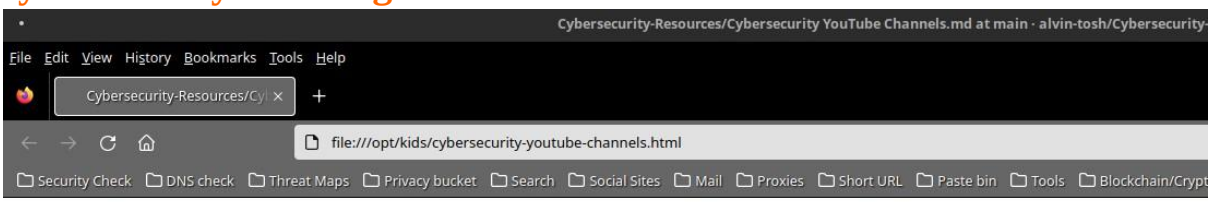
For this purpose, I have made a list of websites you can visit and practice your cyber skills. Every site has a different angle on the whole thing, so I'll summarize in a couple of words its specifics.

Some sites will offer you tutorials to help you, others will require you to find things on your own.

I will update this post below regularly and add sites to this post so bookmark it and/or follow me to see the latest overview.

Site name	Description
$natch competition	Remote banking system containing common vulnerabilities.
Arizona Cyber Warfare Range	The ranges offer an excellent platform for you to learn computer network attack (CNA), computer network defense (CND), and digital forensics (DF). You can
Avatao	More than 350 hands-on challenges (free and paid) to master IT security and it's growing day by day.
Bodgelt Store	The Bodgelt Store is a vulnerable web application which is currently aimed at people who are new to pen testing.
bWAPP	buggy web application, is a free and open source deliberately insecure web application. It helps security enthusiasts, developers and students to discover ar bWAPP prepares one to conduct successful penetration testing and ethical hacking projects.
Cyber Degrees	Free online cyber security Massive Open Online Courses (MOOCS).
Commix testbed	A collection of web pages, vulnerable to command injection flaws.
CryptOMG	CryptOMG is a configurable CTF style test bed that highlights common flaws in cryptographic implementations.
Cyber Security Base	Cyber Security Base is a page with free courses by the University of Helsinki in collaboration with F-Secure.
Cybersecuritychallenge UK	Cyber Security Challenge UK runs a series of competitions designed to test your cyber security skills.
CyberTraining 365	Cybertraining365 has paid material but also offers free classes. The link is directed at the free classes.
Cybrary.it	Free and Open Source Cyber Security Learning.
Damn Small Vulnerable Web	Damn Small Vulnerable Web (DSVW) is a deliberately vulnerable web application written in under 100 lines of code, created for educational purposes. It supp web application vulnerabilities together with appropriate attacks.
Damn Vulnerable Android App	Damn Vulnerable Android App (DVAA) is an Android application which contains intentional vulnerabilities.
Damn Vulnerable Hybrid Mobile App	Damn Vulnerable Hybrid Mobile App (DVHMA) is a hybrid mobile app (for Android) that intentionally contains vulnerabilities.
Damn Vulnerable iOS App	Damn Vulnerable iOS App (DVIA) is an iOS application that is damn vulnerable.
Damn Vulnerable Linux	Damn Vulnerable Linux (DVL) is everything a good Linux distribution isn't. Its developers have spent hours stuffing it with broken, ill-configured, outdated, ar vulnerable to attacks.
Damn Vulnerable Router Firmware	The goal of this project is to simulate a real-world environment to help people learn about other CPU architectures outside of the x86_64 space. This project discovering new things about hardware.
Damn Vulnerable Stateful Web App	Short and simple vulnerable PHP web application that naïve scanners found to be perfectly safe.
Damn Vulnerable Thick Client App	DVTA is a Vulnerable Thick Client Application developed in C# .NET with many vulnerabilities.
Damn Vulnerable Web App	Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application s
Damn Vulnerable Web Services	Damn Vulnerable Web Services is an insecure web application with multiple vulnerable web service components that can be used to learn real-world web se
Damn Vulnerable Web Sockets	Damn Vulnerable Web Sockets (DVWS) is a vulnerable web application which works on web sockets for client-server communication.
Damnvulnerable.me	A deliberately vulnerable modern-day app with lots of DOM-related bugs.
Dareyourmind	Online game, hacker challenge (mirror archive).

Cybersecurity Training Channels

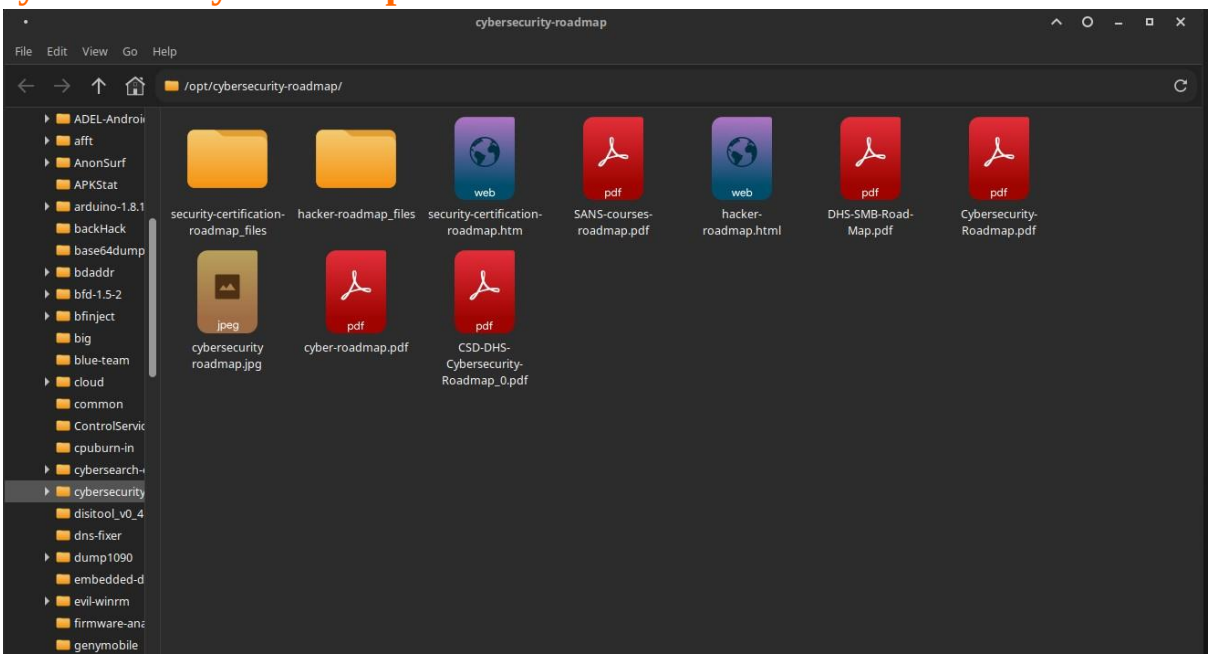


Cybersecurity-Youtube Channels

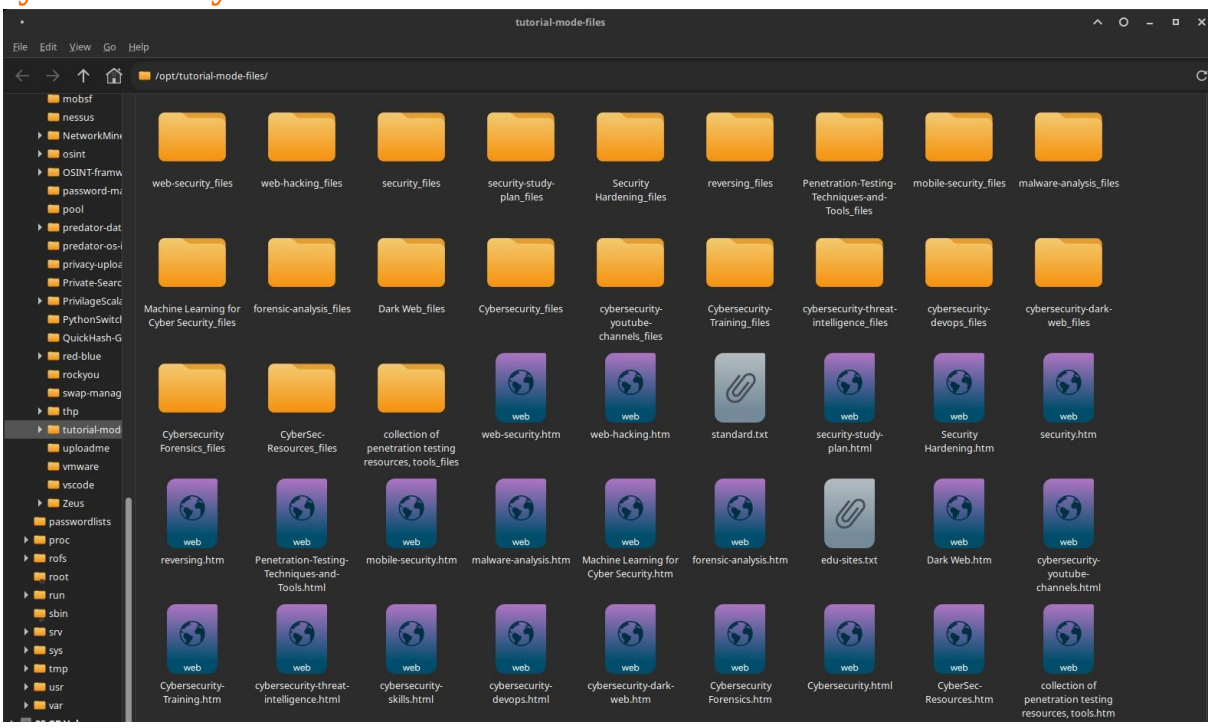
A collection of list of Best Cybersecurity Youtube Channels I have came across and found useful so far

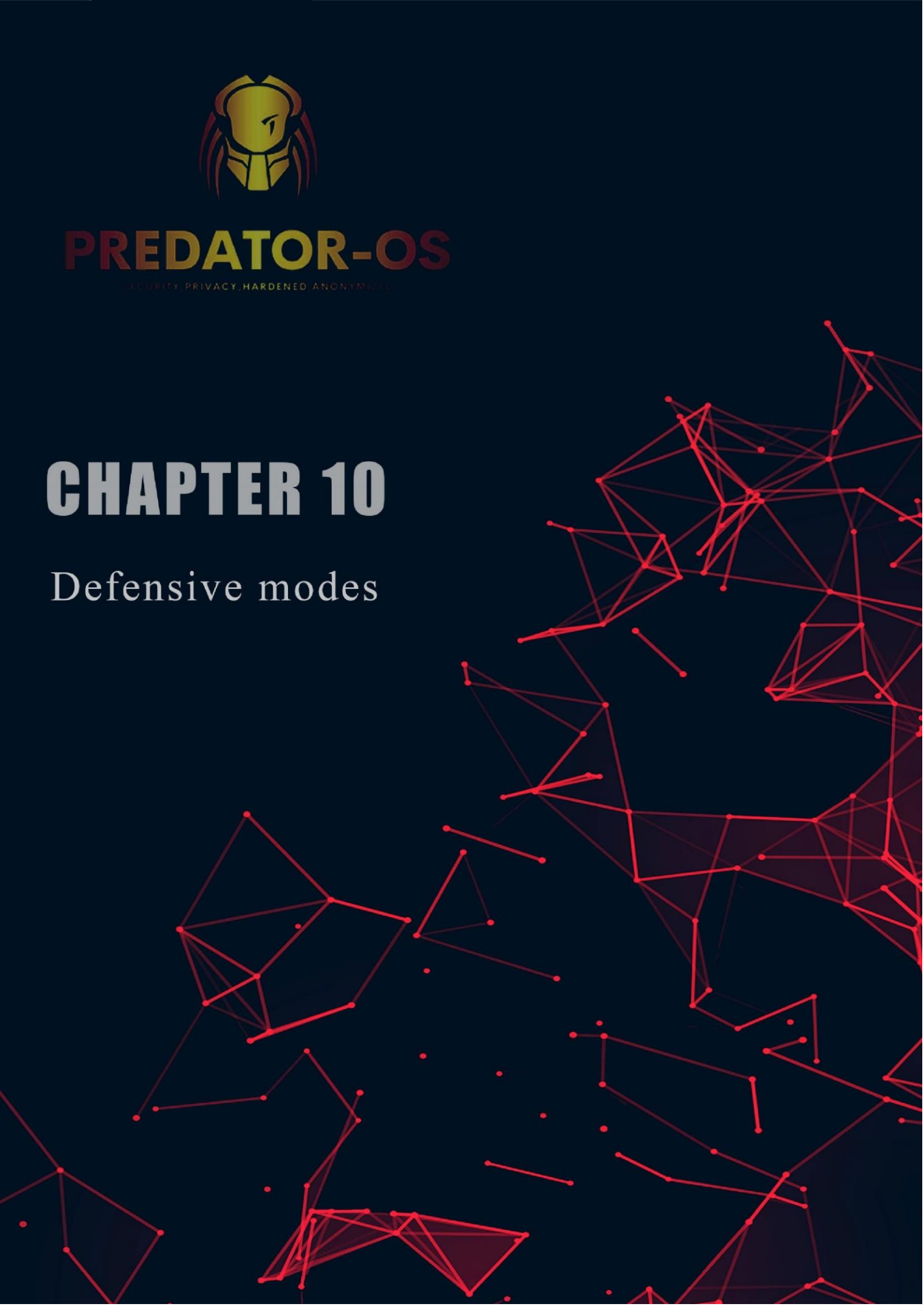
1. InsiderPHD
2. Rana Khalil
3. Spin the hack
4. PwnFunction
5. Cyber Sec Village
6. Farah Hawa
7. Stefan Rows
8. 13Cubed
9. I.T Security Labs
10. Cybr
11. The XSS Rat
12. Cristi Vlad
13. HackerOne
14. PinkDraconian
15. Elevate Cyber
16. Forensic Tech
17. Hak5
18. The cyber mentor
19. Null byte
20. Hackersploit
21. STOK
22. IppSec
23. ScriptKiddieHub - Tadi
24. zSecurity
25. Jon Good
26. Ankit Chauhan
27. Cybersecurity Web
28. 247CTF
29. Motasem Hamdan
30. I.T. Career Questions
31. Hacksplained
32. Bug Bounty Reports Explained
33. TechChip
34. Technical Navigator
35. Beau Knows Tech... Stuff
36. CyberSecurityTV
37. CYBER EVOLUTION
38. David Bombal
39. Nahamsec
40. The cyber expert

Cybersecurity Roadmaps



Cybersecurity Tutorial Files



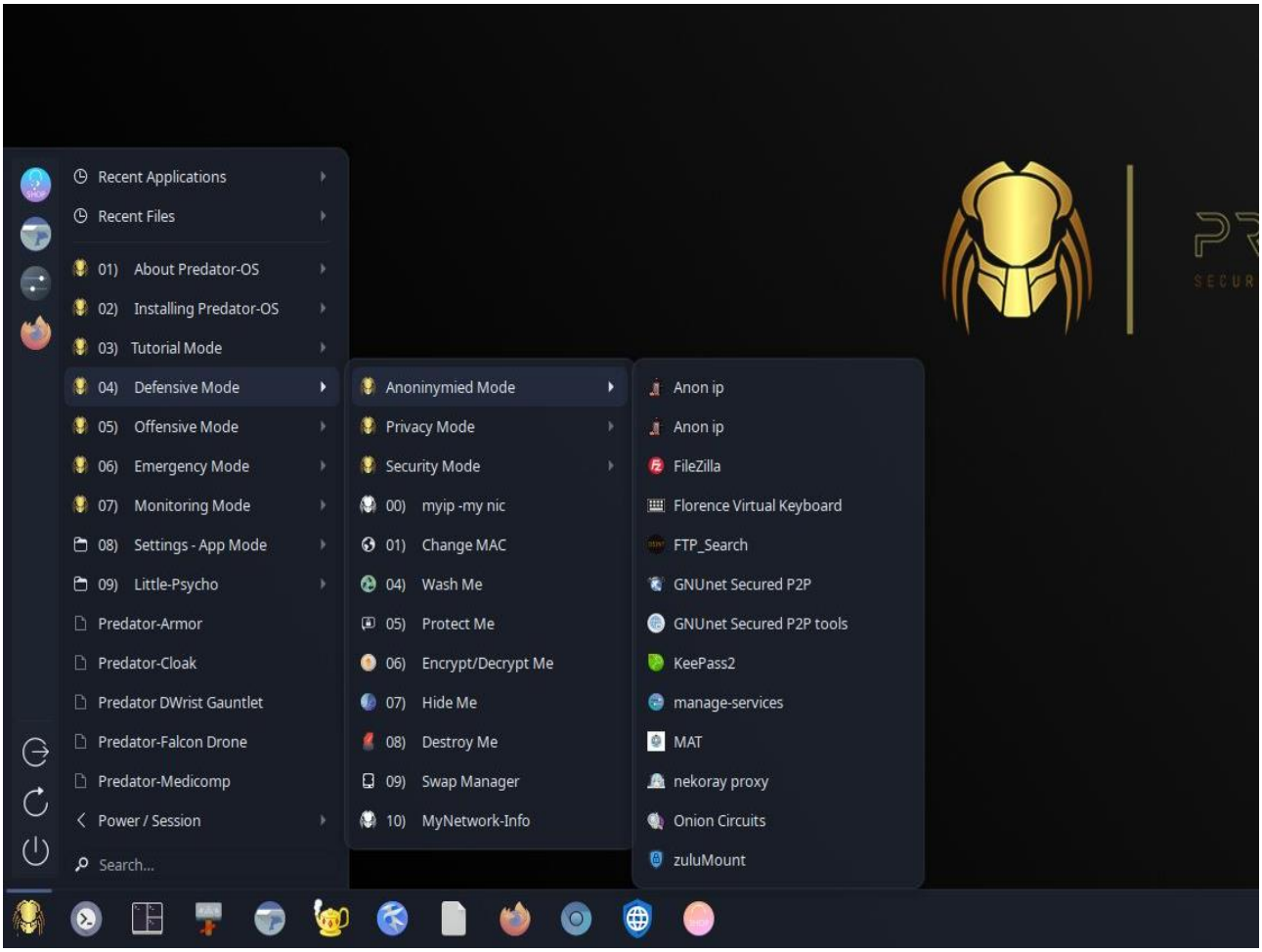


CHAPTER 10

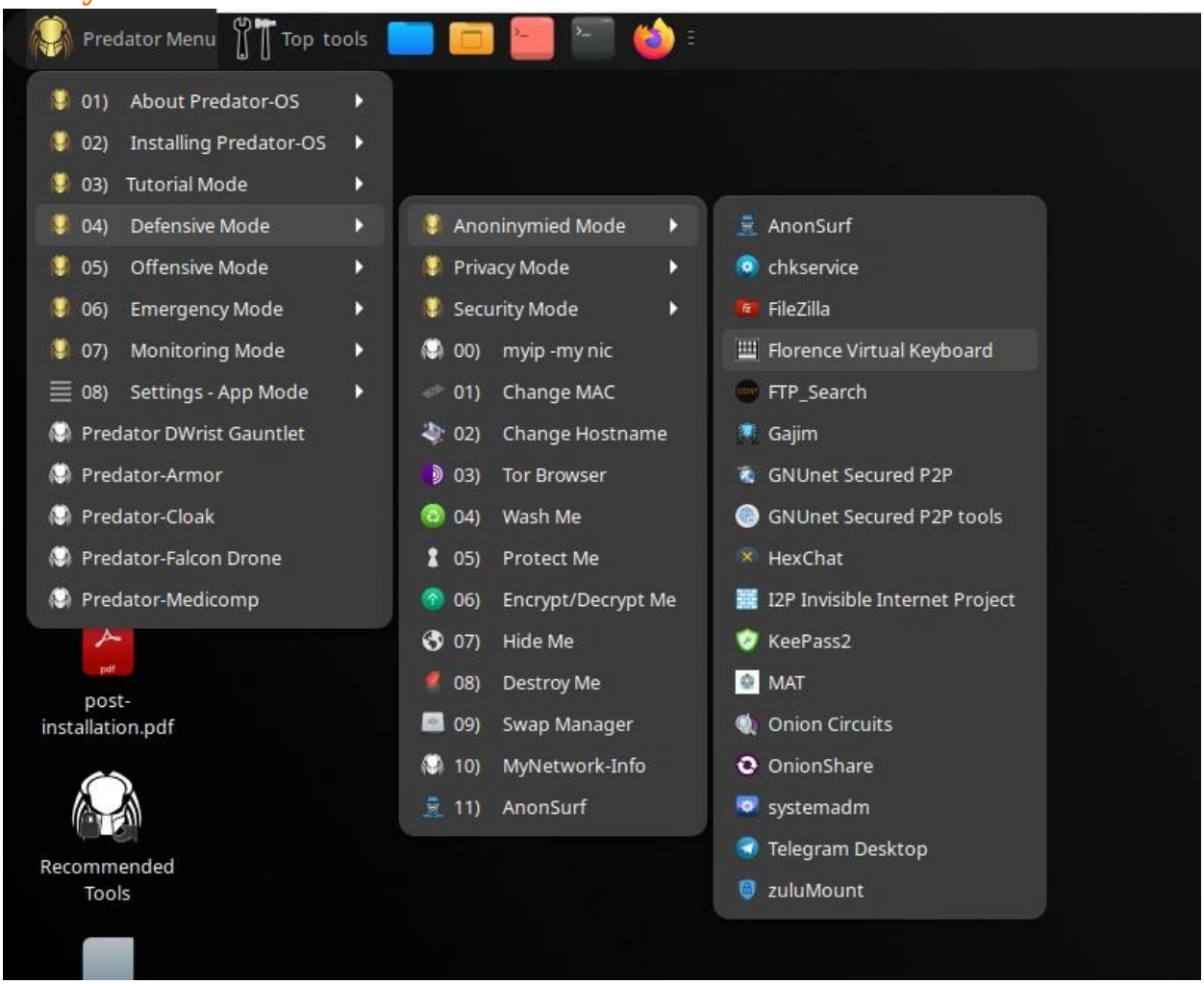
Defensive modes

Chapter 10 Defensive modes

In this mode; which is related to penetration testing and legitimate hacking, the distribution has 1200 pre-installed and configured tools, which are grouped into 40 categories. The layouts are such that simple users can easily access the related tools. In the offensive mode, there are also categories for security students or researchers, including creating test labs and testing malware files, as well as data mining and data analysis tools.



Anonymous modes



Anonymous

Anonymized defensive mode that focuses on the anonymity of the user. Many settings have been made on the kernel and vulnerable programs.

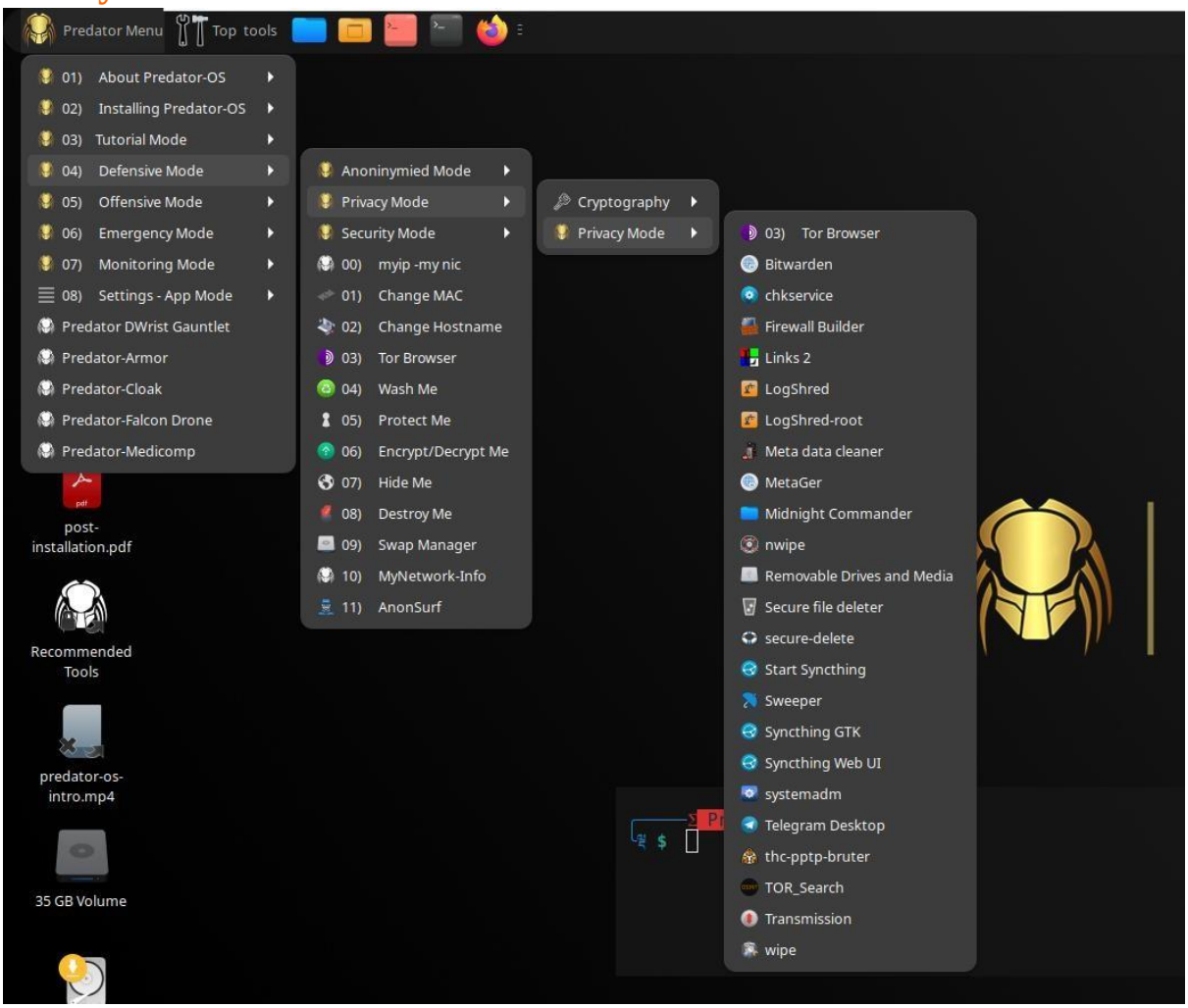
In the privacy protection mode, the focus is on maintaining privacy and protecting users' data and the environment. There are also various tools and useful basic settings for this.

In order to secure the system and the user's activities, the secure defense mode uses various and useful monitoring and protection programs and is installed on the distribution.

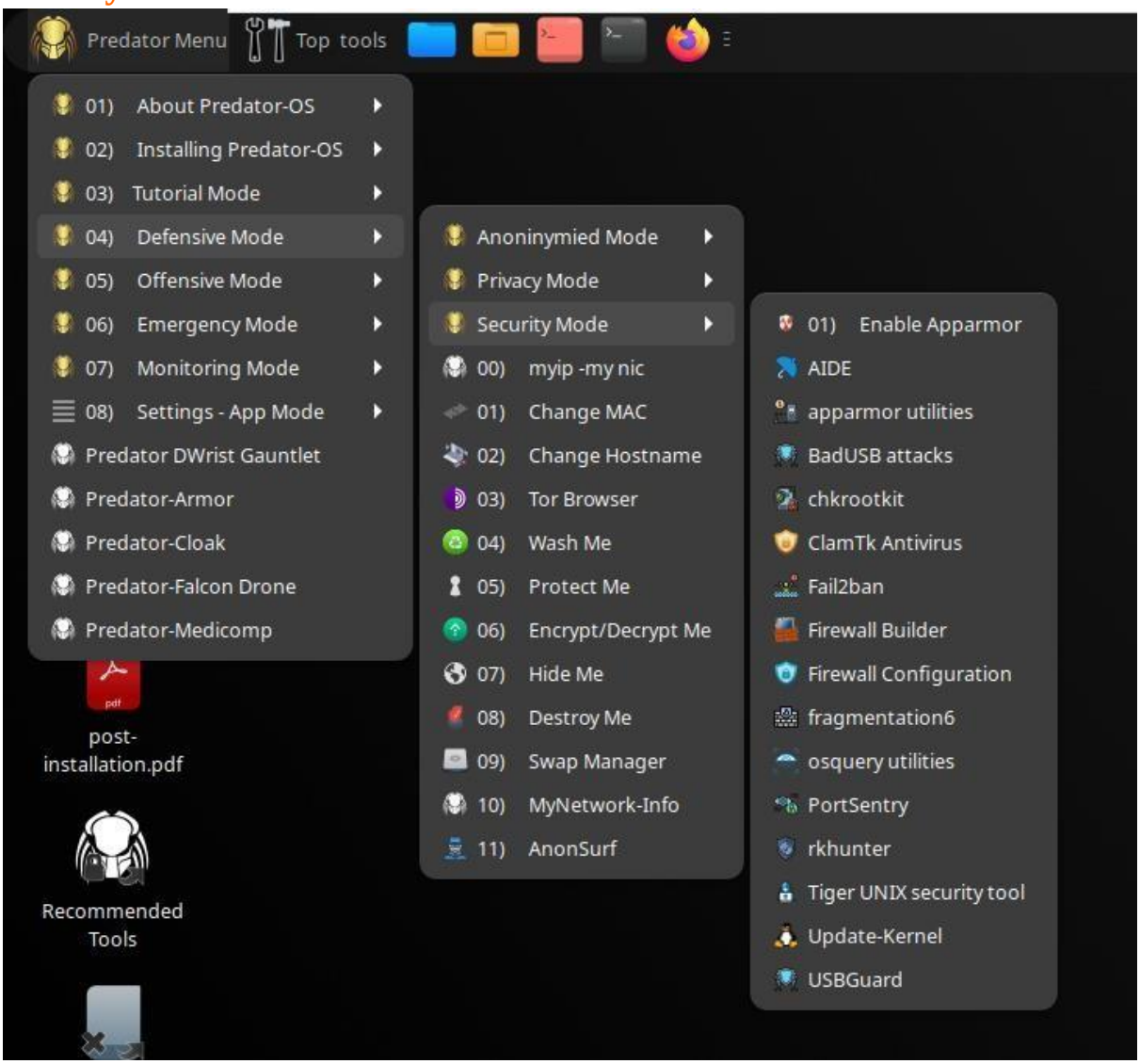
Hardened mode has many settings at the user and kernel levels that protect the distribution against various attacks.

However, the distribution does not work only in the attack mode, and has an attack mode as well. Therefore, many settings related to the defense modes have been avoided in order to avoid problems and interferences and have been left to the user.

Privacy Mode

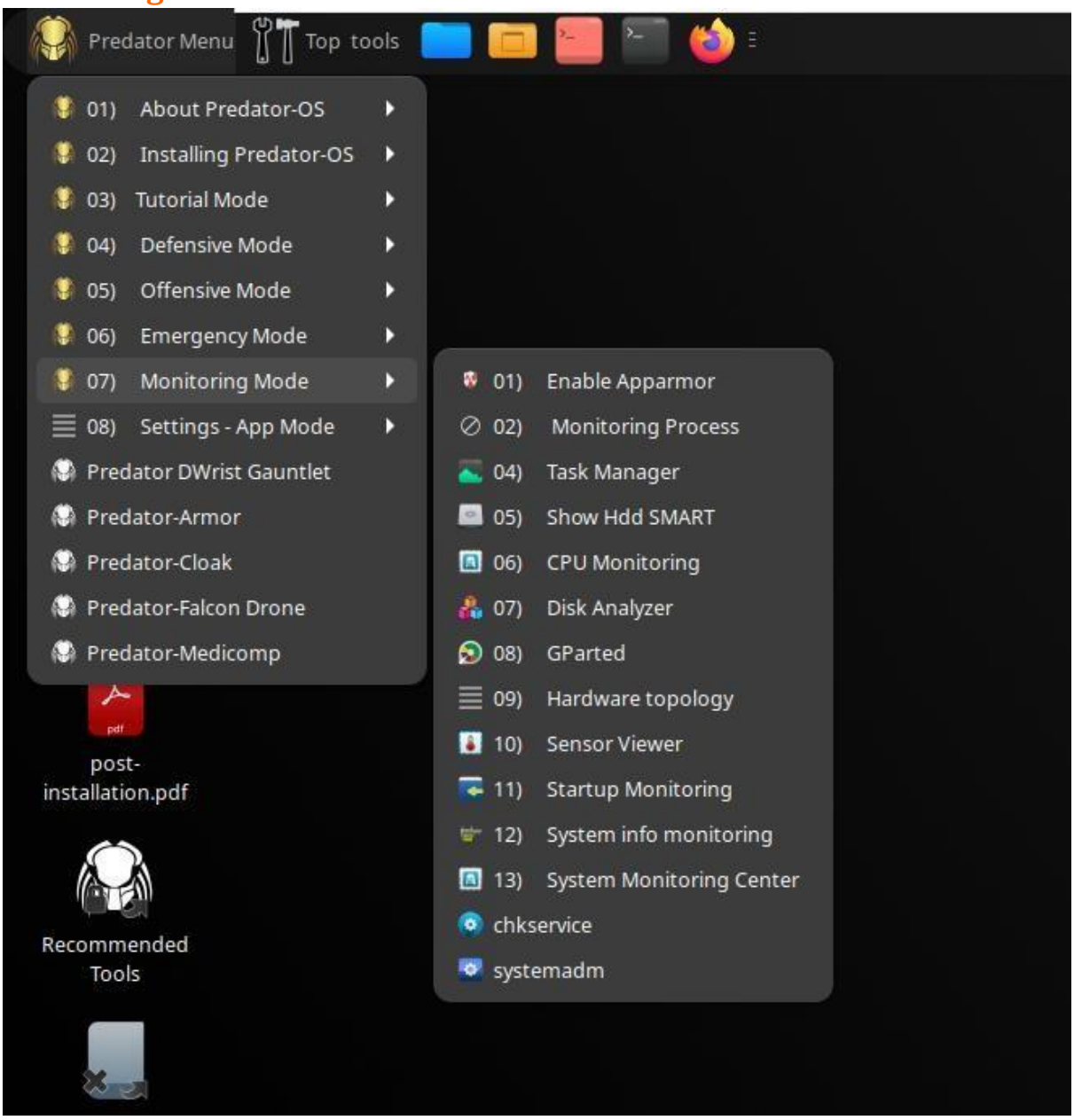


Security Mode

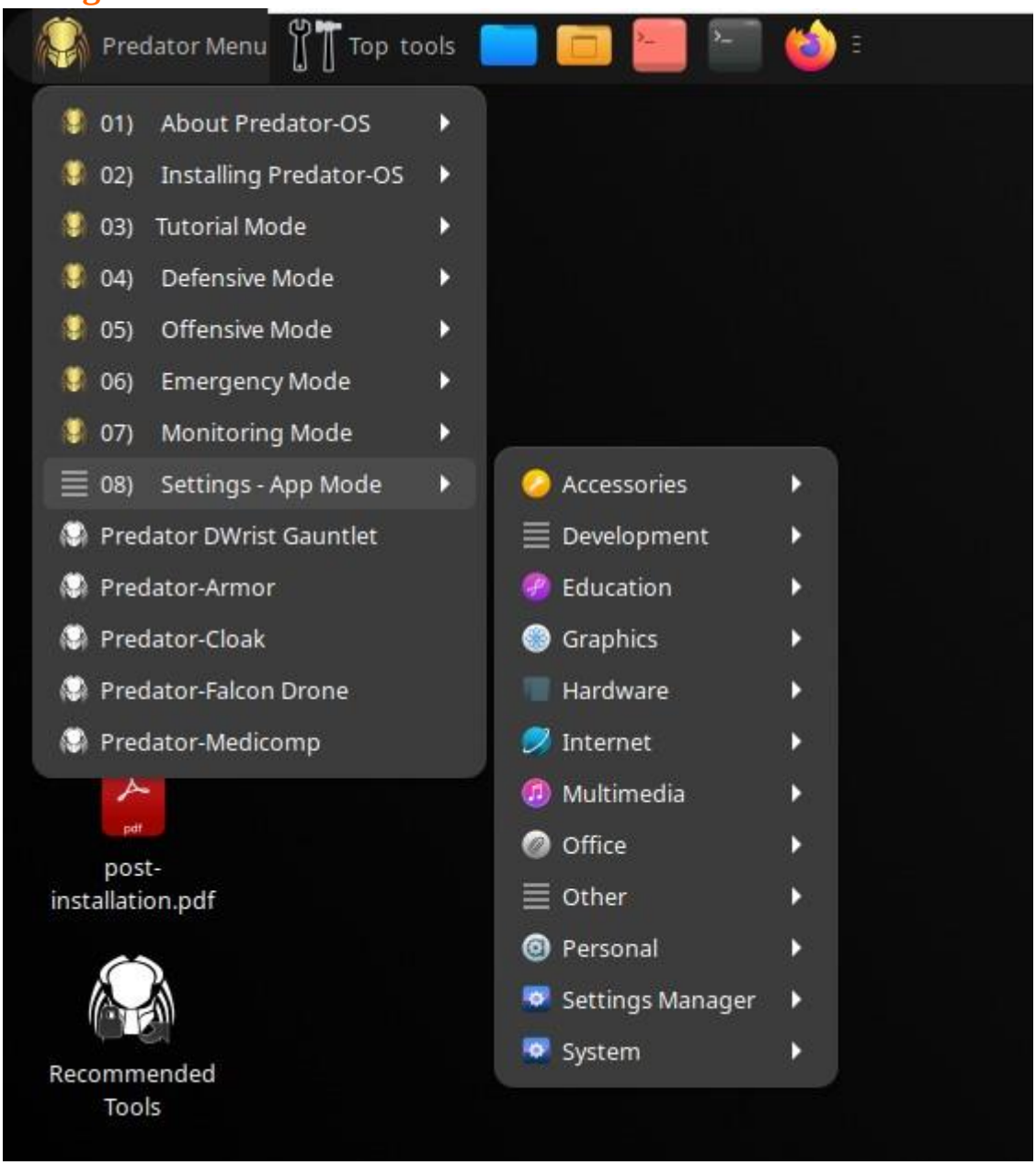


Emergency Mode

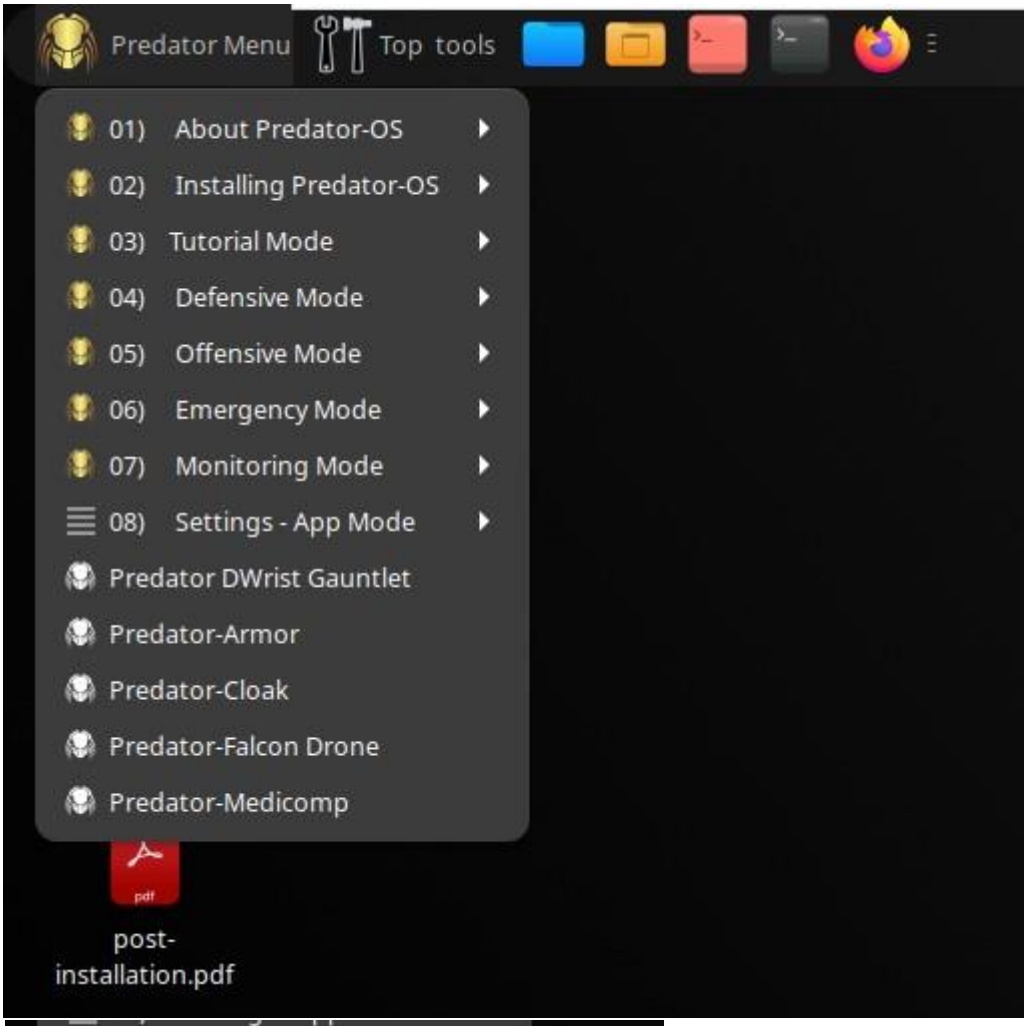
Monitoring Mode



Settings Mode



Dashboard


Predator Menu
Top tools
01) About Predator-OS
02) Installing Predator-OS
03) Tutorial Mode
04) Defensive Mode
05) Offensive Mode
06) Emergency Mode
07) Monitoring Mode
08) Settings - App Mode
Predator DWrist Gauntlet
Predator-Armor
Predator-Cloak
Predator-Falcon Drone
Predator-Medicomp
post-installation.pdf


Predator DWrist Gauntlet
Predator-Armor
Predator-Cloak
Predator-Falcon Drone
Predator-Medicomp
post-installation.pdf

PREDATOR-OS

SECURITY, PRIVACY, HARDENED ANONYMITY

CHAPTER 11

Offensive Mode

Chapter 11

Offensive Mode
Tool Categories:

Osint

Bug Bounty

CTF Tools

Search engine

Virtual Lab Setup

Android Emulator

Remote Tools

Mount Manager

Computer Forensics

Digital Forensic

Carving Tools

Image Forensic

Pdf Forensic

Recovery Tools

Keylogger

Security Auditing

Enmuration

Log Manager

Privilege Escalation

Vulnerability Assessment

Car Hacking

Risk assessment

Exploit Tools

Post Exploit

Security Loopholes

Fuzzing Tools

Stress Testing

Payload Tools

Maintance Accessing

Network Pentesting

Active Directory

Wireless Hacking

Transfer Layer

Bluetooth Tools

Rfid Radio Frequency Identifier

Nfc Near Field Communication

Radio Hacking

Sniffing Tools

Spoofing Tools

Cisco Tools

DNS Analysis

Ids/Ips Tools

Live Host Tools

Viop Tools

IPv6 Tools

Tunneling and Exfit

Ping Tools

Dos Attack

Honeypot Tools

Web Attack

Cms Tools

Web Frameworks

Bruteforce

Web Application Tools

Vulnerability

Web Scanners

Cloud Pentesting

IOT Pentesting

Embedded Pentesting

Voip Pentesting

Flash Disk Manipulation

Os Attack Tools

Mobile Pentesting

Hardware Pentesting

Lock Picking Tools

Cryptology

Password List

Reversing Engineer

Idea Tools

Editor and Hex Tools

Debuggers Tools

Disassembler Tools Compiler Tools

Malware Analysis Kit

Backdoor Tools

Cybersecurity Machine Learning

PREDATOR-OS

SECURITY, PRIVACY, HARDENED ANONYMITY

CHAPTER 12

Configurations

Chapter 12

Predator-OS Configuration

Configuring the System for another language

If the system was installed using French, the machine will probably already have French set as the default language. However, it is good to know what the installer does to set the language, so that later, if the need arises, you can change it.

***TOOL* The locale command to display the current configuration**

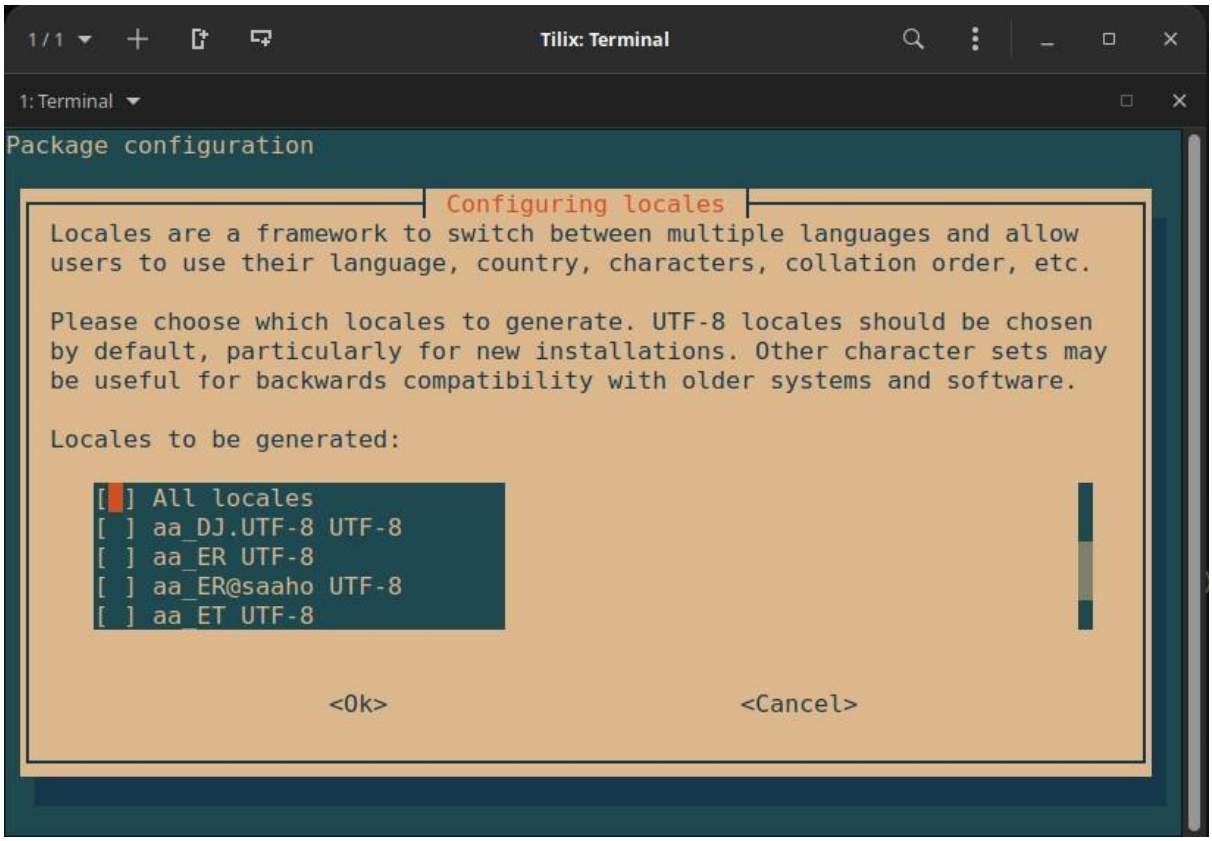
The **locale** command lists a summary of the current configuration of various locale parameters (date format, numbers format, etc.), presented in the form of a group of standard environment variables dedicated to the dynamic modification of these settings.

Setting the Default Language

A locale is a group of regional settings. This includes not only the language for text, but also the format for displaying numbers, dates, times, and monetary sums, as well as the alphabetical comparison rules. Although each of these parameters can be specified independently from the others, we generally use a locale, which is a coherent set of values for these parameters corresponding to a "region" in the broadest sense. These locales are usually indicated under the form, **language-code_COUNTRY-CODE**, sometimes with a suffix to specify the character set and encoding to be used. This enables consideration of idiomatic or typographical differences between different regions with a common language.

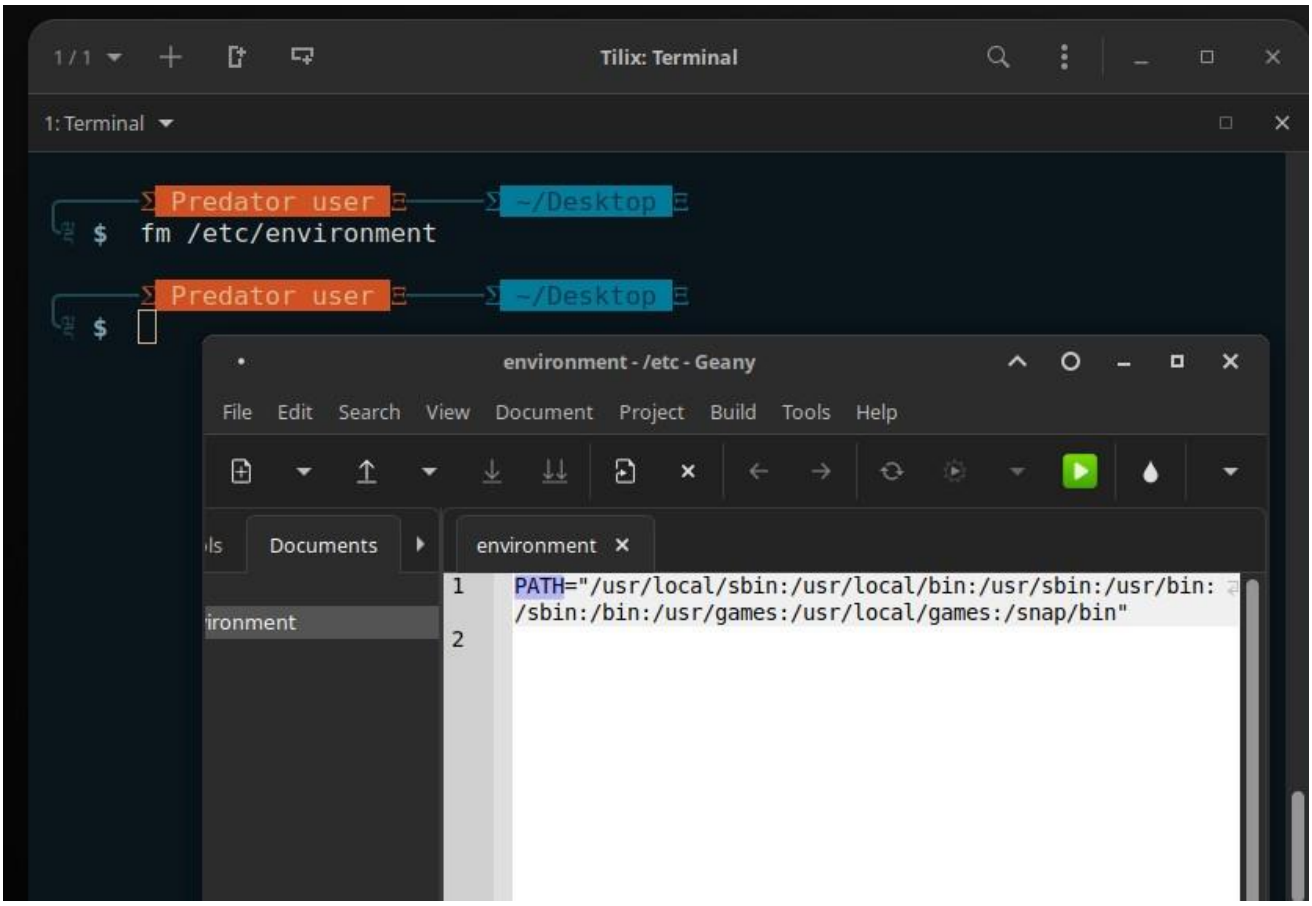
The locales package includes all the elements required for proper functioning of "localization" of various applications. During installation, this package will ask you to select a set of supported languages. This set can be changed at any time by running **dpkg-reconfigure locales** as root.

```
# dpkg-reconfigure locales
```

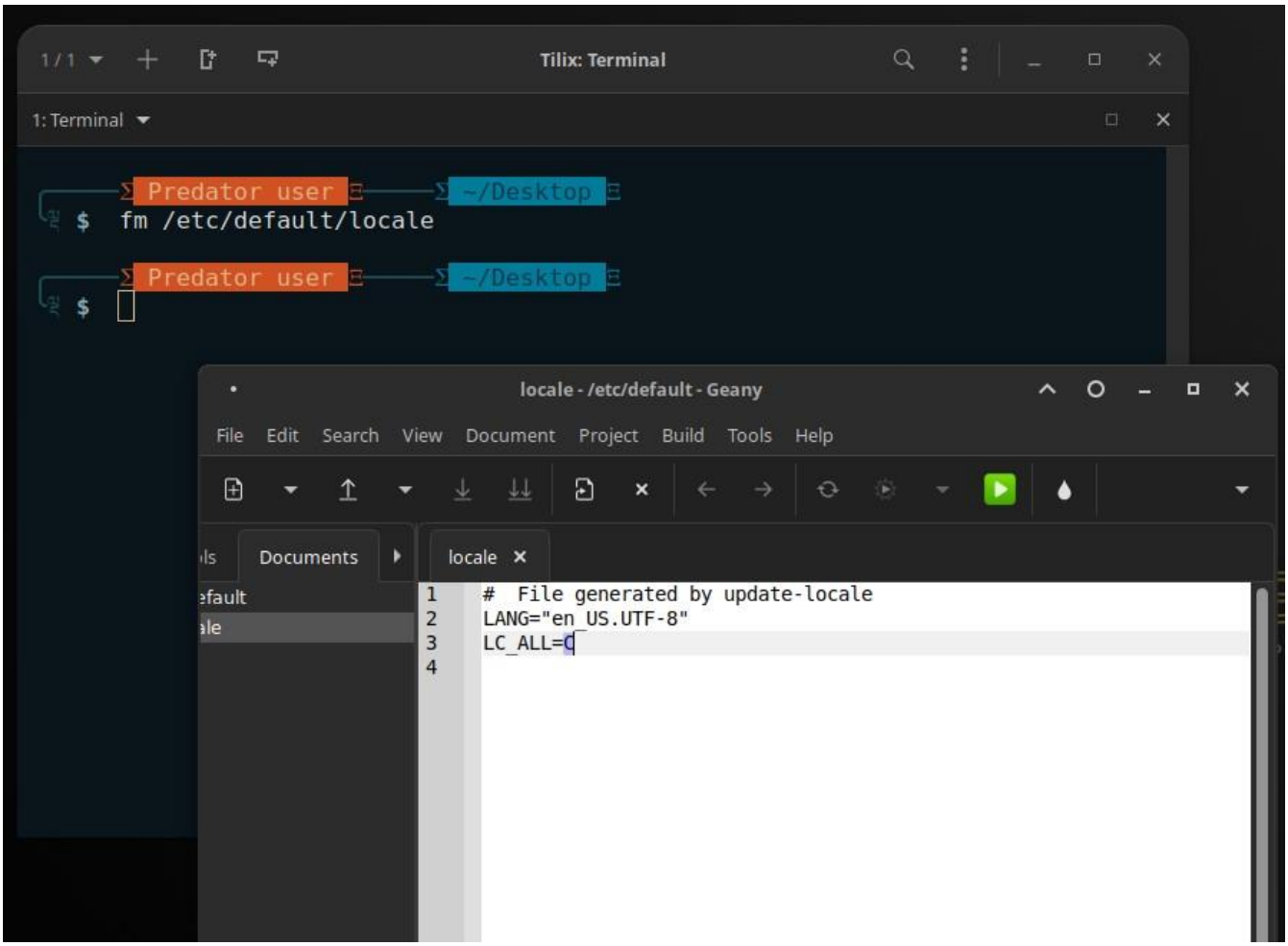


The first question invites you to select "locales" to support. Selecting all English locales (meaning those beginning with "**en_**") is a reasonable choice. Do not hesitate to also enable other locales if the machine will host foreign users. The list of locales enabled on the system is stored in the **/etc/locale.gen** file. It is possible to edit this file by hand, but you should run **locale-gen** after any modifications. It will generate the necessary files for the added locales to work, and remove any obsolete files.
The second question, entitled "Default locale for the system environment", requests a default locale. The recommended choice in the USA is "**en_US.UTF-8**". British English speakers will prefer "**en_GB.UTF-8**", and Canadians will prefer either "**en_CA.UTF-8**" or, for French, "**fr_CA.UTF-8**". The **/etc/default/locale** file will then be modified to store this choice. From there, it is picked up by all user sessions since PAM will inject its content in the **LANG** environment variable.

The **/etc/environment** file provides the **login**, **gdm**, or even **ssh** programs with the correct environment variables to be created.



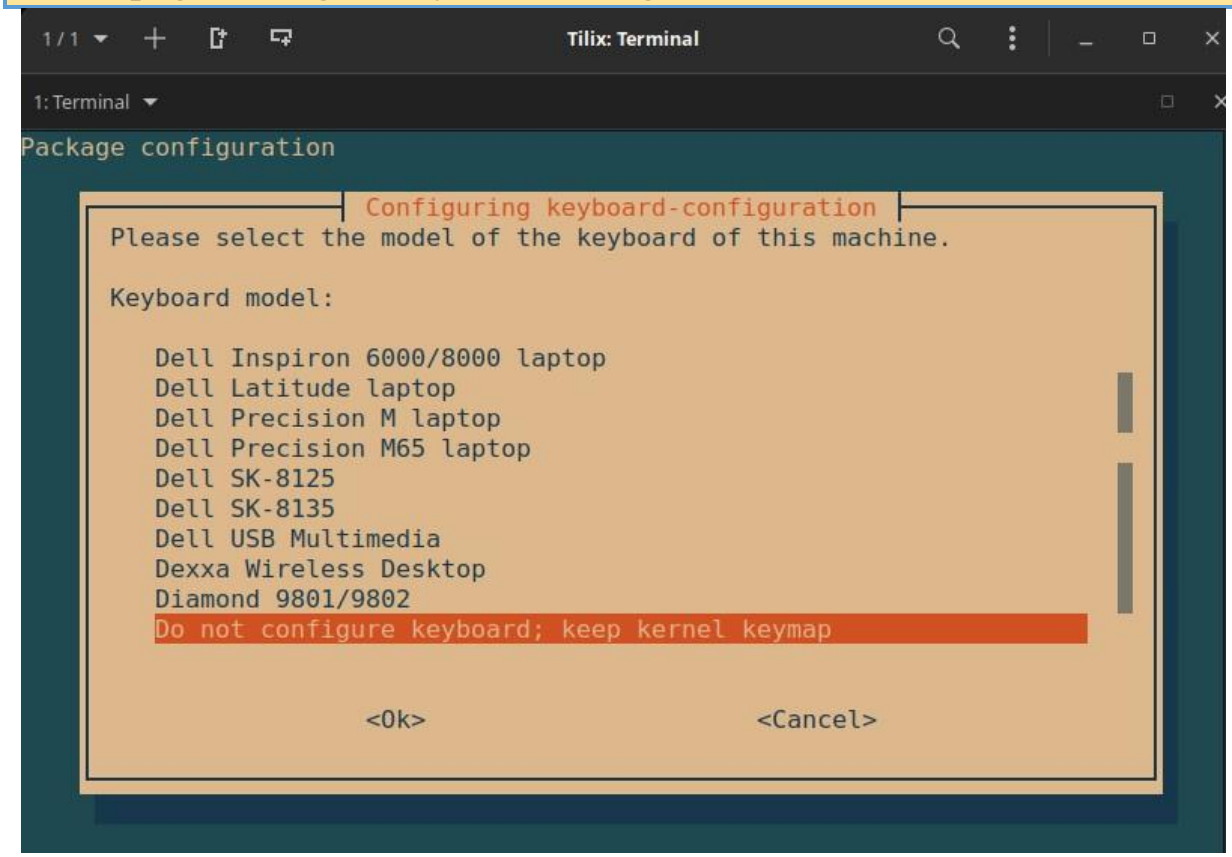
The /etc/default/locale file works in a similar manner, but contains only the LANG environment variable. Thanks to this split, some PAM users can inherit a complete environment without localization. Indeed, it is generally discouraged to run server programs with localization enabled; on the other hand, localization and regional settings are recommended for programs that open user sessions.


1 / 1
Tilix: Terminal
1: Terminal
Predator user
~/Desktop
$ fm /etc/default/locale
Predator user
~/Desktop
$
locale - /etc/default - Geany
File Edit Search View Document Project Build Tools Help
Documents
locale
1 # File generated by update-locale
2 LANG="en_US.UTF-8"
3 LC_ALL=C
4

Configuring the Keyboard

Even if the keyboard layout is managed differently in console and graphical mode, it offers a single configuration interface that works for both: it is based on debconf and is implemented in the keyboard-configuration package. Thus the **dpkgreconfigure keyboard-configuration** command can be used at any time to reset the keyboard layout.

```
$sudo dpkg-reconfigure keyboard-configuration
```

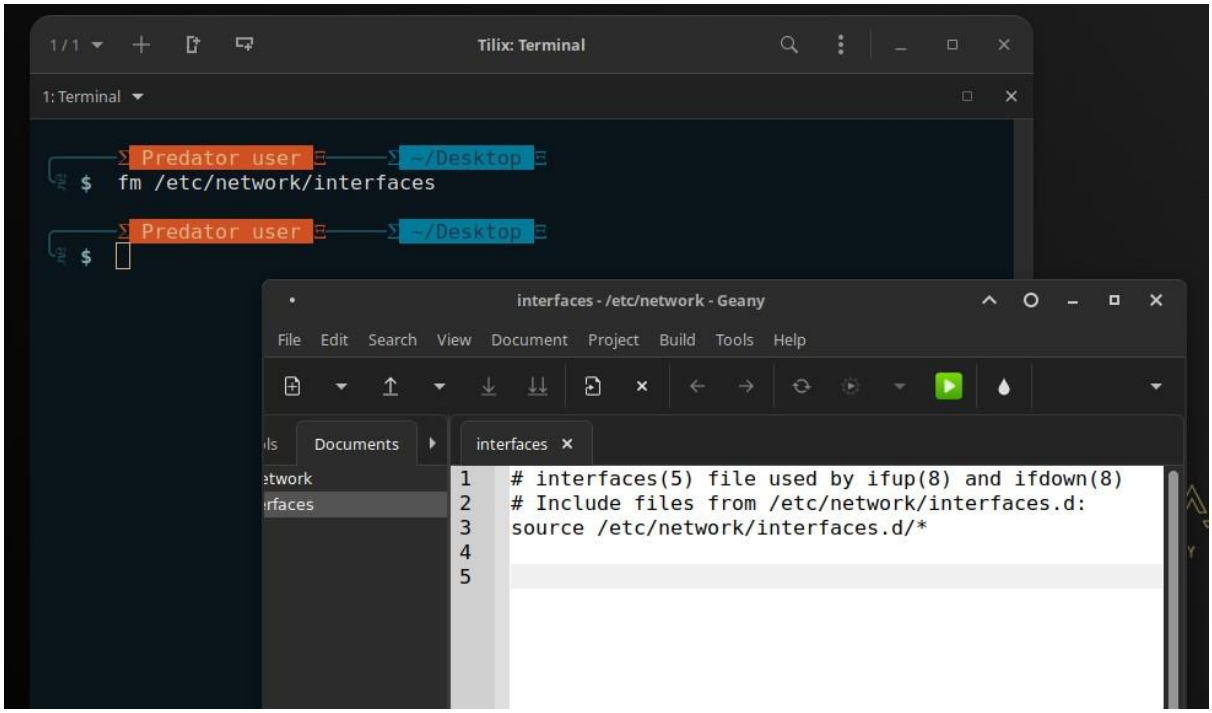


Migrating to UTF-8

The generalization of UTF-8 encoding has been a long-awaited solution to numerous difficulties with interoperability, since it facilitates international exchange and removes the arbitrary limits on characters that can be used in a document. The one drawback is that it had to go through a rather difficult transition phase. Since it could not be completely transparent (that is, it could not happen at the same time all over the world), two conversion operations were required: one on file contents, and the other on filenames. Fortunately, the bulk of this migration has been completed and we discuss it largely for reference.

Configuring the Network

If Network Manager is not installed, then the installer will configure ifupdown by creating the **/etc/network/interfaces** file. A line starting with **auto** gives a list of interfaces to be automatically configured on boot by the **networking** service. When there are many interfaces, it is good practice to keep the configuration in different files inside **/etc/network/interfaces.d/** as described in sidebar BACK TO BASICS Directories ending in **.d**.

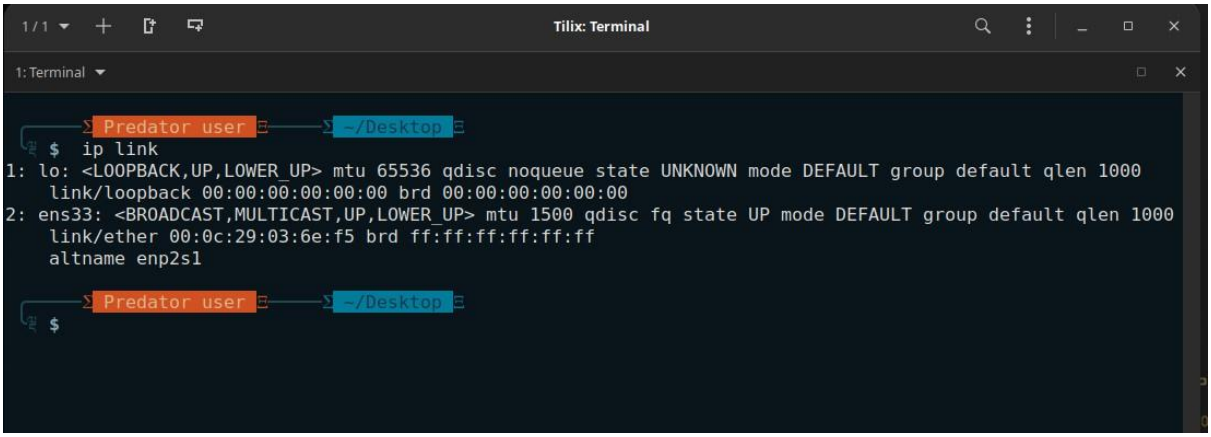


Ethernet Interface

If the computer has an Ethernet card, the IP network that is associated with it must be configured by choosing from one of two methods. The simplest method is dynamic configuration with DHCP, and it requires a DHCP server on the local network. It may indicate a desired hostname, corresponding to the **hostname** setting in the example below. The DHCP server then sends configuration settings for the appropriate network.

Names of network interfaces

By default, the kernel attributes generic names such as **eth0** (for wired Ethernet) or **wlan0** (for WiFi) to the network interfaces. The number in those names is a simple incremental counter representing the order in which they have been detected. With modern hardware, that order (at least in theory) might change for each reboot and thus the default names are not reliable.



Wireless Interface

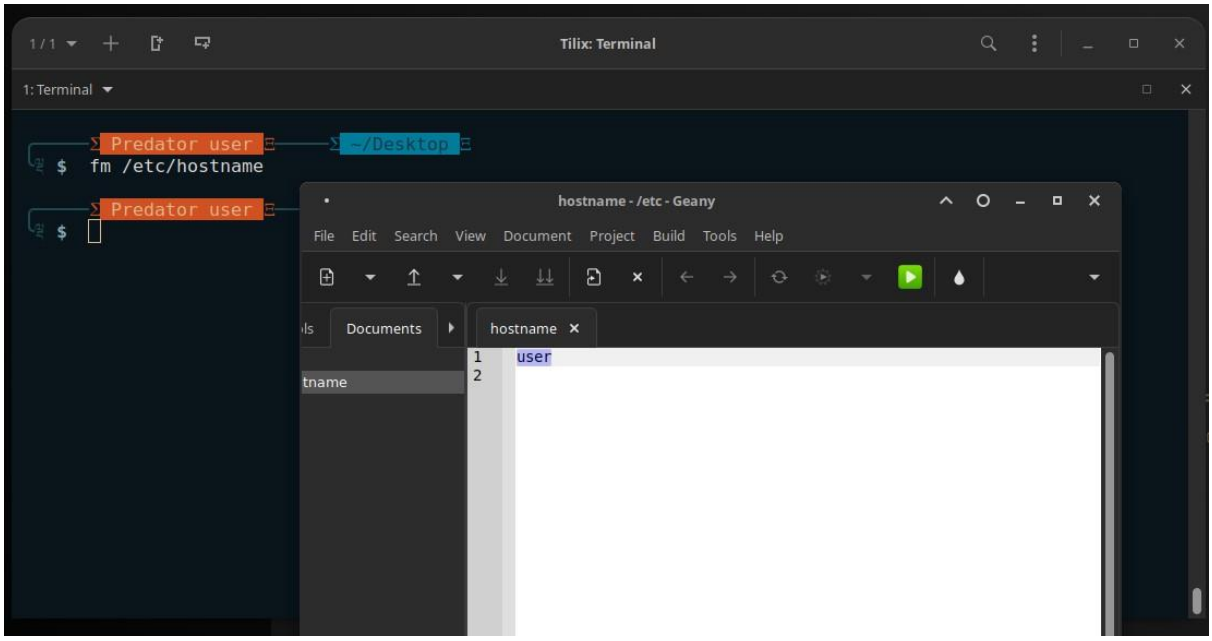
Getting wireless network cards to work can be a bit more challenging. First of all, they often require the installation of proprietary firmwares which are not installed by default. Then wireless networks rely on cryptography to restrict access to authorized users only, this implies storing some secret key in the network configuration. Let's tackle those topics one by one.

Network Manager knows how to handle various types of connections (DHCP, manual configuration, local network), but only if the configuration is set with the program itself. This is why it will systematically ignore all network interfaces in **/etc/network/interfaces** and **/etc/network/interfaces.d/** for which it is not suited. Since Network Manager does not give details when no network connections are shown, the easy way is to delete from **/etc/network/interfaces** any configuration for all interfaces that must be managed by Network Manager.

Setting the Hostname and Configuring the Name Service

The purpose of assigning names to IP numbers is to make them easier for people to remember. In reality, an IP address identifies a network interface associated with a device such as a network card. Since each machine can have several network cards, and several interfaces on each card, one single computer can have several names in the domain name system.

Each machine is, however, identified by a main (or "canonical") name, stored in the **/etc/hostname** file and communicated to the Linux kernel by initialization scripts through the **hostname** command. The current value is available in a virtual filesystem, and you can get it with the **cat /proc/sys/kernel/hostname** command.

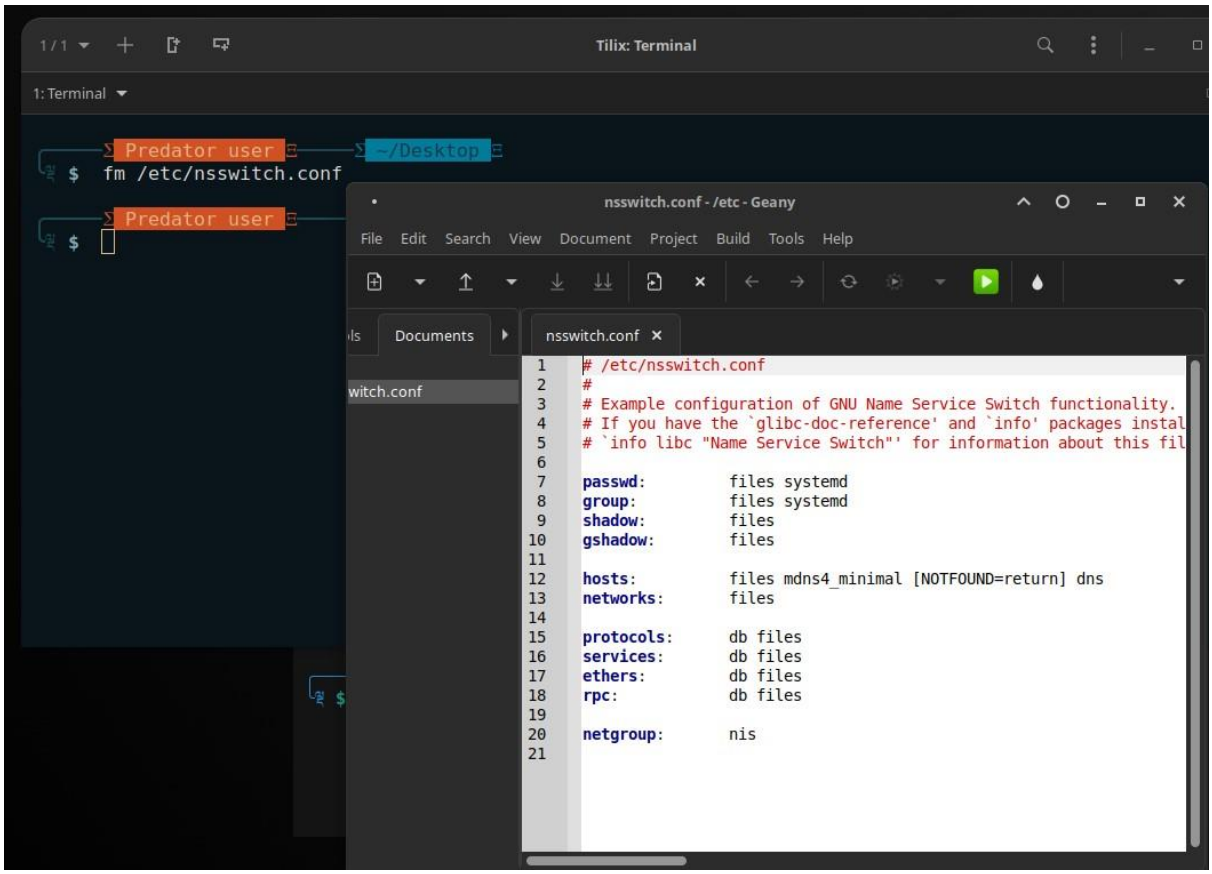


Surprisingly, the domain name is not managed in the same way, but comes from the complete name of the machine, acquired through name resolution. You can change it in the **/etc/hosts** file; simply write a complete name for the machine there at the beginning of the list of names associated with the address of the machine, as in the following example:

```
127.0.0.1    localhost
```

Name Resolution

The mechanism for name resolution in Linux is modular and can use various sources of information declared in the **/etc/nsswitch.conf** file. The entry that involves host name resolution is **hosts**. By default, it contains **files dns**, which means that the system consults the **/etc/hosts** file first, then DNS servers. NIS/NIS+ or LDAP servers are other possible sources.



Configuring DNS Servers

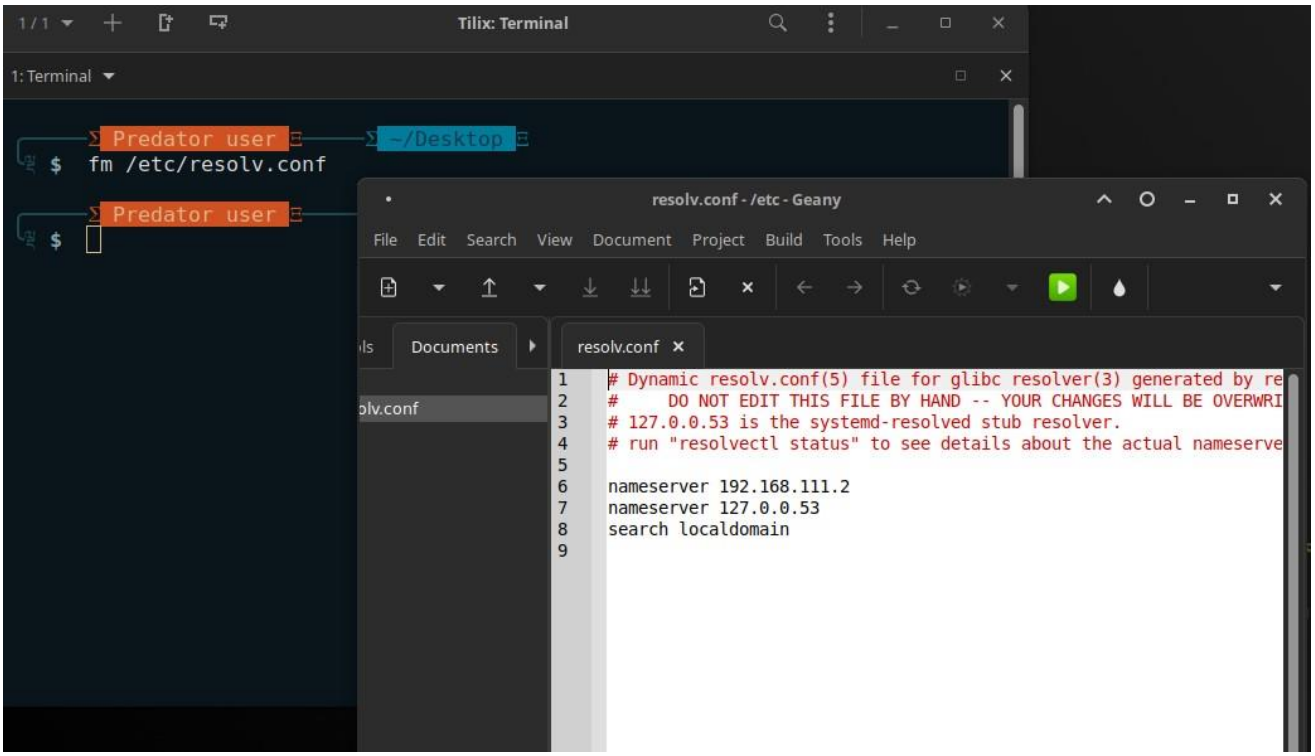
DNS (Domain Name Service) is a distributed and hierarchical service mapping names to IP addresses, and vice-versa. Specifically, it can turn a human-friendly name such as **www.Predator-OS.com** into the actual IP address, **213.244.11.247**.

To access DNS information, a DNS server must be available to relay requests. Falcot Corp has its own, but an individual user is more likely to use the DNS servers provided by their ISP.

The DNS servers to be used are indicated in **/etc/resolv.conf**, one per line, with the **nameserver** keyword preceding an IP address, as in the following example:

```
nameserver 212.27.32.176 nameserver
212.27.32.177 nameserver 8.8.8.8
```

Note that the **/etc/resolv.conf** file may be handled automatically (and overwritten) when the network is managed by NetworkManager or configured via DHCP, or when resolvconf is installed or systemd-resolved(8) is enabled.



The /etc/hosts file

If there is no name server on the local network, it is still possible to establish a small table mapping IP addresses and machine hostnames in the **/etc/hosts** file, usually reserved for local network stations. The syntax of this file as described in hosts(5) is very simple: each line indicates a specific IP address followed by the list of any associated names (the first being "completely qualified", meaning it includes the domain name).

This file is available even during network outages or when DNS servers are unreachable, but will only really be useful when duplicated on all the machines on the network. The slightest alteration in correspondence will require the file to be updated everywhere. This is why **/etc/hosts** generally only contains the most important entries.This file will be sufficient for a small network not connected to the Internet, but with 5 machines or more, it is recommended to install a proper DNS server.

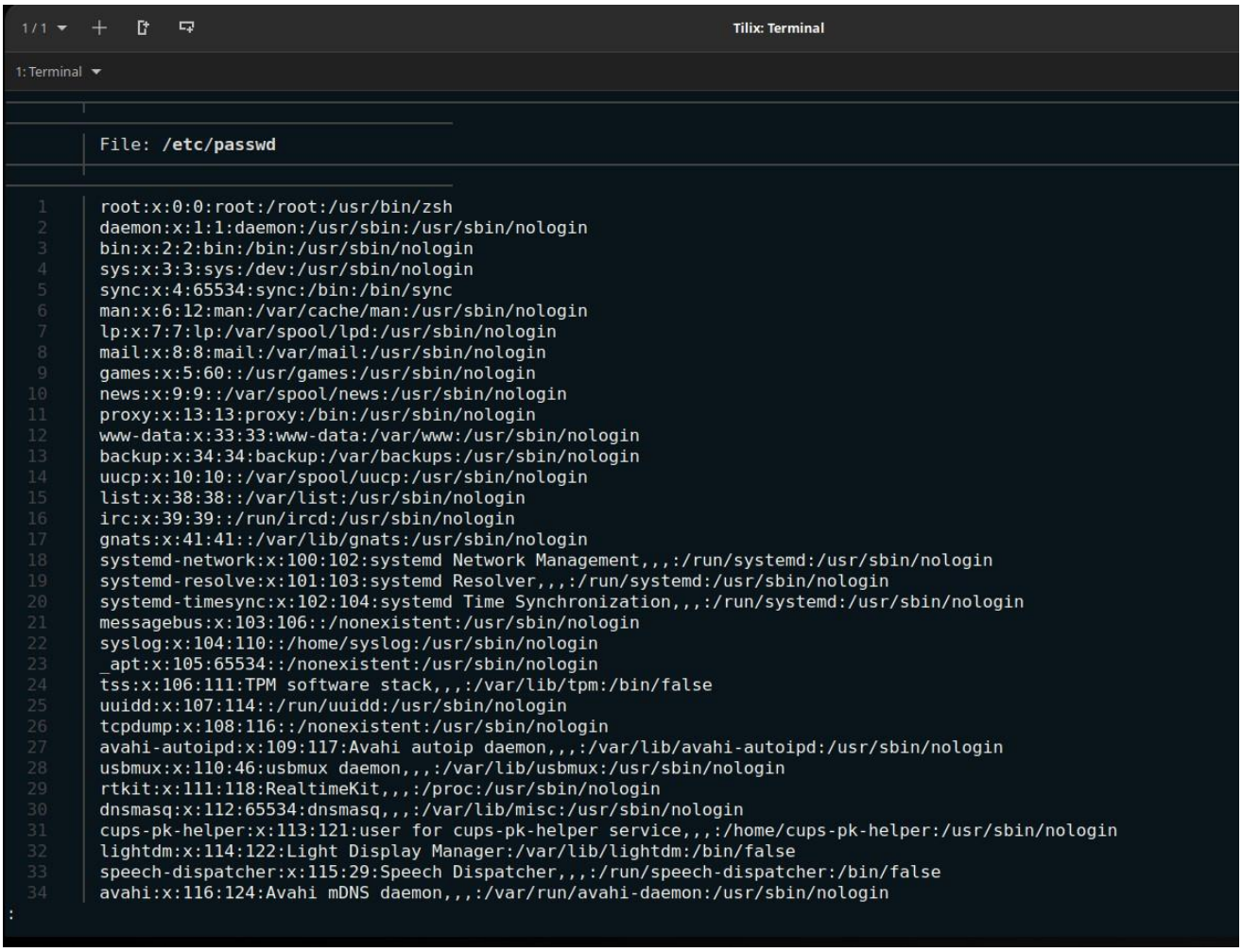
User and Group Databases

The list of users is usually stored in the **/etc/passwd** file, while the **/etc/shadow** file stores hashed passwords. Both are text files, in a relatively simple format, which can be read and modified with a text editor. Each user is listed there on a line with several fields separated with a colon (":").

Users and groups are used on GNU/Linux for **access control**—that is, to control access to the system's files, directories, and peripherals. Linux offers relatively simple/coarse access control mechanisms by default.

Sudoers file

The /etc/sudoers file contains a list of users or user groups with permission to execute a subset of commands while having the privileges of the root user or another specified user. The program may be configured to require a password.



User List: /etc/passwd

Here is the list of fields in the /etc/passwd file:

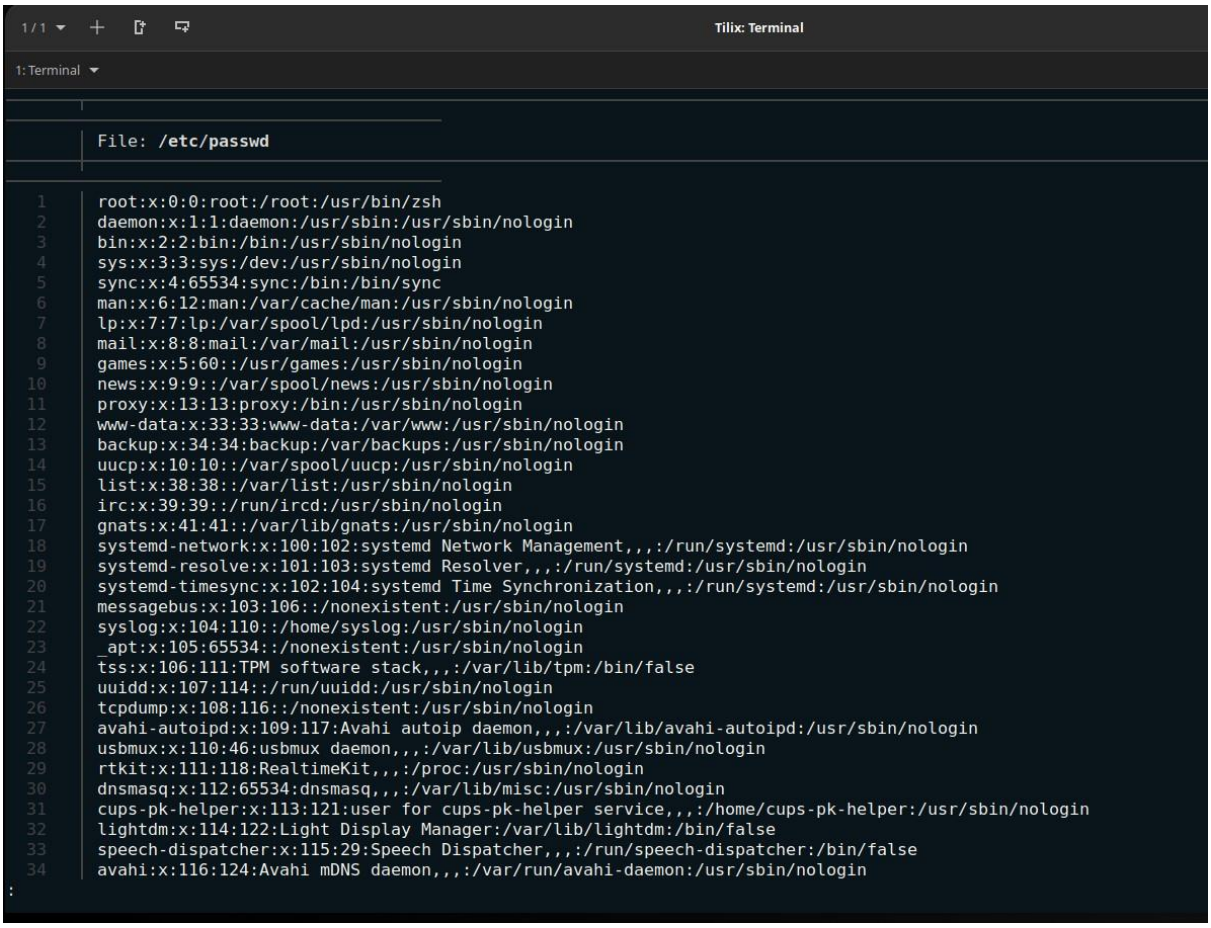
- login, for example **rhertzog**;
- **password**: this is a password encrypted by a one-way function (**crypt**), relying on **DES**, **MD5**, **SHA-256** or **SHA-512**. The special value "**x**" indicates that the encrypted password is stored in /etc/shadow;
- **uid**: unique number identifying each user;
- **gid**: unique number for the user's main group
- **GECOS**: data field usually containing the user's full name;
- login directory, assigned to the user for storage of their personal files (the environment variable $**HOME** generally points here);
- program to execute upon login. This is usually a command interpreter (shell), giving the user free rein. If you specify /bin/false (which does nothing and returns control immediately), the user cannot login.

The Linux /etc/shadow file

On Linux, the shadow password file is readable only by the superuser and serves to keep encrypted passwords safe from prying eyes and password cracking programs. It also includes some additional account information that wasn't provided for in the original **/etc/passwd** format. These days, shadow passwords are the default on all systems.

The **shadow** file is not a superset of the **passwd** file, and the **passwd** file is not generated from it. You must maintain both files or use tools such as **useradd** that maintain both files on your behalf. Like **/etc/passwd**, **/etc/shadow** contains one line for each user. Each line contains nine fields, separated by colons:

- Login name
- Encrypted password
- Date of last password change
- Minimum number of days between password changes
- Maximum number of days between password changes
- Number of days in advance to warn users about password expiration
- Days after password expiration that account is disabled
- Account expiration date
- A field reserved for future use which is currently always empty



The Hidden and Encrypted Password File: /etc/shadow The **/etc/shadow** file contains the following fields:

- login;
- encrypted password;
- Several fields managing password expiration.

One can expire passwords using this file or set the time until the account is disabled after the password has expired.

Modifying an Existing Account or Password

The following commands allow modification of the information stored in specific fields of the user databases: **passwd** permits a regular user to change their password, which in turn, updates the **/etc/shadow** file (**chpasswd** allows administrators to update passwords for a list of users in batch mode); **chfn** (CHange Full Name), reserved for the super-user (root), modifies the **GECOS** field. **chsh** (CHange SHell) allows the user to change their login shell; however, available choices will be limited

to those listed in **/etc/shells**; the administrator, on the other hand, is not bound by this restriction and can set the shell to any program of their choosing.

Finally, the **chage** (CHange AGE) command allows the administrator to change the password expiration settings (the **-l user** option will list the current settings). You can also force the expiration of a password using the **passwd -e user** command, which will require the user to change their password the next time they log in.

Besides these tools the **usermod** command allows to modify all the details mentioned above.

Setting a password

Set a password for a new user with

```
$ sudo passwd newusername
```

You will be prompted for the actual password.

Some automated systems for adding new users do not require you to set an initial password. Instead, they force the user to set a password on first login. Although this feature is convenient, it's a giant security hole: anyone who can guess new login names (or look them up in **/etc/passwd**) can swoop down and hijack accounts before the intended users have had a chance to log in.

Disabling an Account

You may find yourself needing to "disable an account" (lock out a user), as a disciplinary measure, for the purposes of an investigation, or simply in the event of a prolonged or definitive absence of a user. A disabled account means the user cannot login or gain access to the machine. The account remains intact on the machine and no files or data are deleted; it is simply inaccessible.

Disabling the root account

If your site standardizes on the use of **sudo**, you will have surprisingly little use for actual root passwords. Most of your administrative team will never have occasion to use them.

That fact raises the question of whether a root password is necessary at all. If you decide that it isn't, you can disable root logins entirely by setting root's encrypted password to * or to some other fixed, arbitrary string. On Linux, **passwd -l** "locks" an account by prepending a ! to the encrypted password, with equivalent results. The * and the ! are just conventions; no software checks for them explicitly. Their effect derives from their not being valid password hashes. As a result, attempts to verify root's password simply fail.

The main effect of locking the root account is that root cannot log in even on the console. Neither can any user successfully run **su**, because that requires a root password check as well. However the root account continues to exist, and all the

software that usually runs as root continues to do so. In particular, **sudo** works normally.

The main advantage of disabling the root account is that you needn't record and manage root's password. You're also eliminating the possibility of the root password being compromised, but that's more a pleasant side effect than a compelling reason to go passwordless. Rarely used passwords are already at low risk of violation.

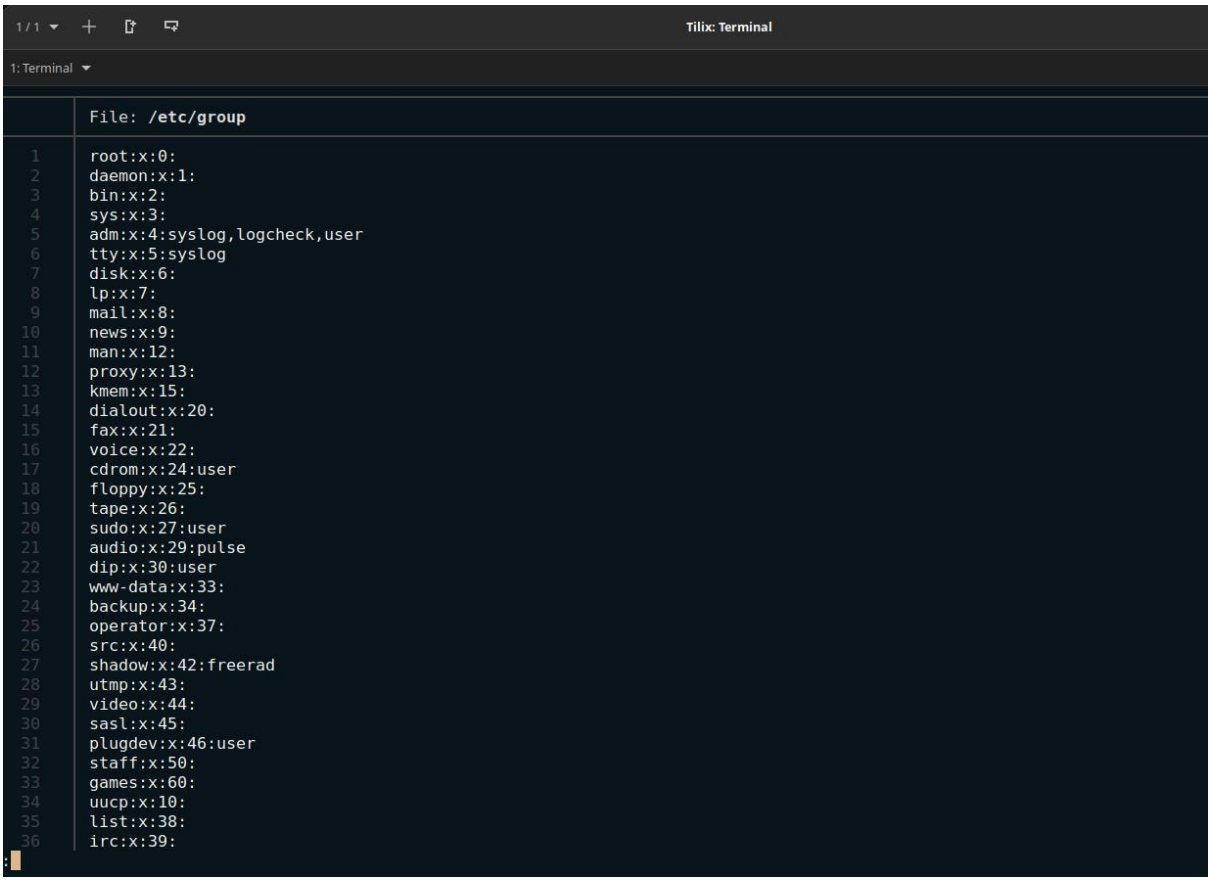
It's particularly helpful to have a real root password on physical computers. Real computers are apt to require rescuing when hardware or configuration problems interfere with **sudo** or the boot process. In these cases, it's nice to have the traditional root account available as an emergency fallback.

Debian stable ships with the root account locked, and all administrative access is funneled through **sudo** or a GUI equivalent. If you prefer, it's fine to set a root password on Debian stable and then unlock the account with **sudo passwd -u root**.

Group List: /etc/group

Groups are listed in the **/etc/group** file, a simple textual database in a format similar to that of the **/etc/passwd** file, with the following fields:

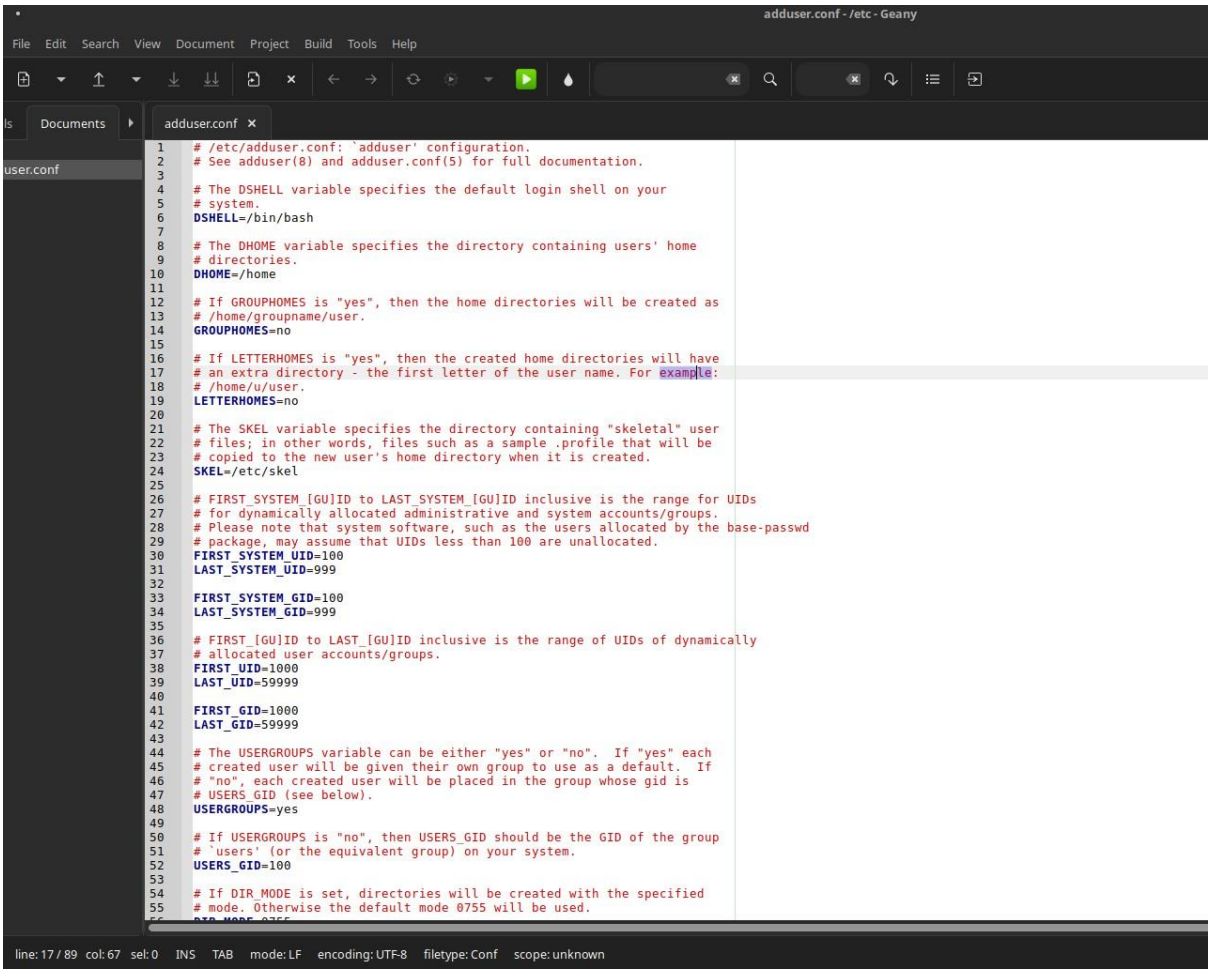
- group name;
- password (optional): This is only used to join a group when one is not a usual member (with the newgrp or sg commands, see sidebar BACK TO BASICS Working with several groups);
- gid: unique group identification number;
- list of members: list of names of users who are members of the group, separated by commas.



Creating Accounts

One of the first actions an administrator needs to do when setting up a new machine is to create user accounts. This is typically done using the **adduser** command which takes a user-name for the new user to be created, as an argument.

The **adduser** command asks a few questions before creating the account, but its usage is straightforward. Its configuration file, **/etc/adduser.conf**, includes all the interesting settings: it can be used to automatically set a quota for each new user by creating a user template, or to change the location of user accounts; the latter is rarely useful, but it comes in handy when you have a large number of users and want to divide their accounts over several disks, for instance. You can also choose a different default shell.



Predator configurations for new user accout

/etc/adduser.conf: `adduser' configuration.

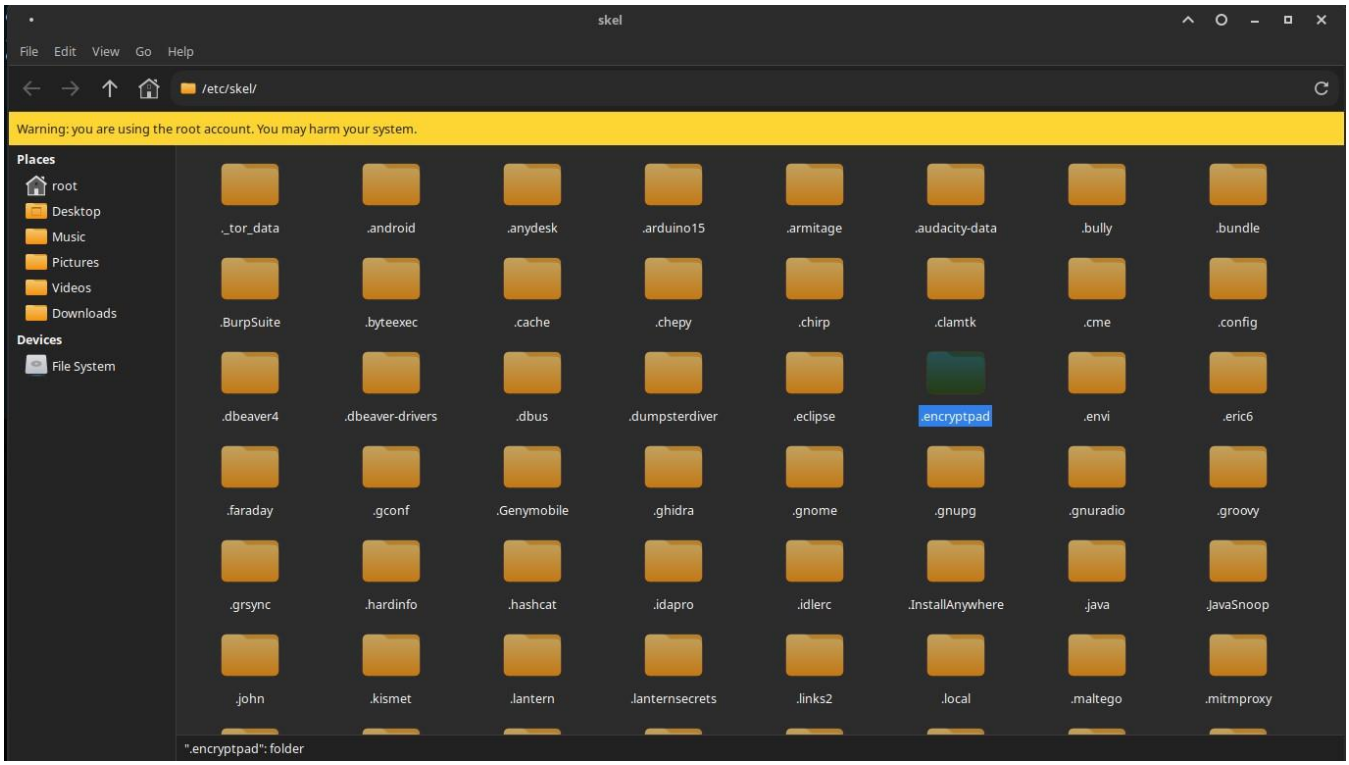
```
# See adduser(8) and adduser.conf(5) for full documentation.

# The DSHELL variable specifies the default login shell on your #
system.
DSHELL=/bin/bash

# The DHOME variable specifies the directory containing users' home #
directories.
DHOME=/home

# If GROUPHOMES is "yes", then the home directories will be created as #
/home/groupname/user.
GROUPHOMES=no

# If LETTERHOMES is "yes", then the created home directories will
have # an extra directory - the first letter of the user name. For example: #
/home/u/user.
```

```
LETTERHOMES=no

# The SKEL variable specifies the directory containing "skeletal" user
# files; in other words, files such as a sample .profile that will be #
copied to the new user's home directory when it is created.
SKEL=/etc/skel

# FIRST_SYSTEM_[GU]ID to LAST_SYSTEM_[GU]ID inclusive is the range for UIDs
# for dynamically allocated administrative and system accounts/groups.
# Please note that system software, such as the users allocated by the base-passwd #
package, may assume that UIDs less than 100 are unallocated.
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=999

FIRST_SYSTEM_GID=100
LAST_SYSTEM_GID=999

# FIRST_[GU]ID to LAST_[GU]ID inclusive is the range of UIDs of dynamically #
allocated user accounts/groups.
FIRST_UID=1000
LAST_UID=59999

FIRST_GID=1000
LAST_GID=59999

# The USERGROUPS variable can be either "yes" or "no". If "yes" each
# created user will be given their own group to use as a default. If
# "no", each created user will be placed in the group whose gid is #
USERS_GID (see below).
USERGROUPS=yes

# If USERGROUPS is "no", then USERS_GID should be the GID of the group #
`users' (or the equivalent group) on your system.
USERS_GID=100

# If DIR_MODE is set, directories will be created with the specified #
mode. Otherwise the default mode 0755 will be used.
DIR_MODE=0755
```

If SETGID_HOME is “yes” home directories for users with their own

```
# group the setgid bit will be set. This was the default for
# versions << 3.13 of adduser. Because it has some bad side effects we
# no longer do this per default. If you want it nevertheless you can #
still set it here.
SETGID_HOME=no

# If QUOTAUSER is set, a default quota will be set from that user with
# `edquota -p QUOTAUSER newuser'
QUOTAUSER=""

# If SKEL_IGNORE_REGEX is set, adduser will ignore files matching this
# regular expression when creating a new home directory
SKEL_IGNORE_REGEX="dpkg-(old|new|dist|save)"

# Set this if you want the --add_extra_groups option to adduser to add #
new users to other groups.
# This is the list of groups that new non-system users will be added to #
Default:
#EXTRA_GROUPS="dialout cdrom floppy audio video plugdev users"

# If ADD_EXTRA_GROUPS is set to something non-zero, the EXTRA_GROUPS
# option above will be default behavior for adding new, non-system users
#ADD_EXTRA_GROUPS=1



# check user and group names also against this regular expression.
#NAME_REGEX="^[a-z][-a-z0-9_]*\$"

# use extrausers by default
#USE_EXTRAUSERS=1
```

The creation of an account populates the user's home directory with the contents of the **/etc/skel/** template. This provides the user with a set of standard directories and configuration files.



Creating the home directory and installing startup files

useradd and adduser create new users' home directories for you, but you will likely want to double-check the permissions and startup files for new accounts. There's nothing magical about home directories. If you neglected to include a home directory when setting up a new user, you can create it with a simple mkdir. You need to set ownerships and permissions on the new directory as well, but this is most efficiently done after you've installed any local startup files.

Startup files traditionally begin with a dot and end with the letters rc, short for "run command," a relic of the CTSS operating system. The initial dot causes ls to hide these "uninteresting" files from directory listings unless the -a option is used.

We recommend that you include default startup files for each shell that is popular on your systems so that users continue to have a reasonable default environment even if they change shells. Table 8.2 lists a variety of common startup files.

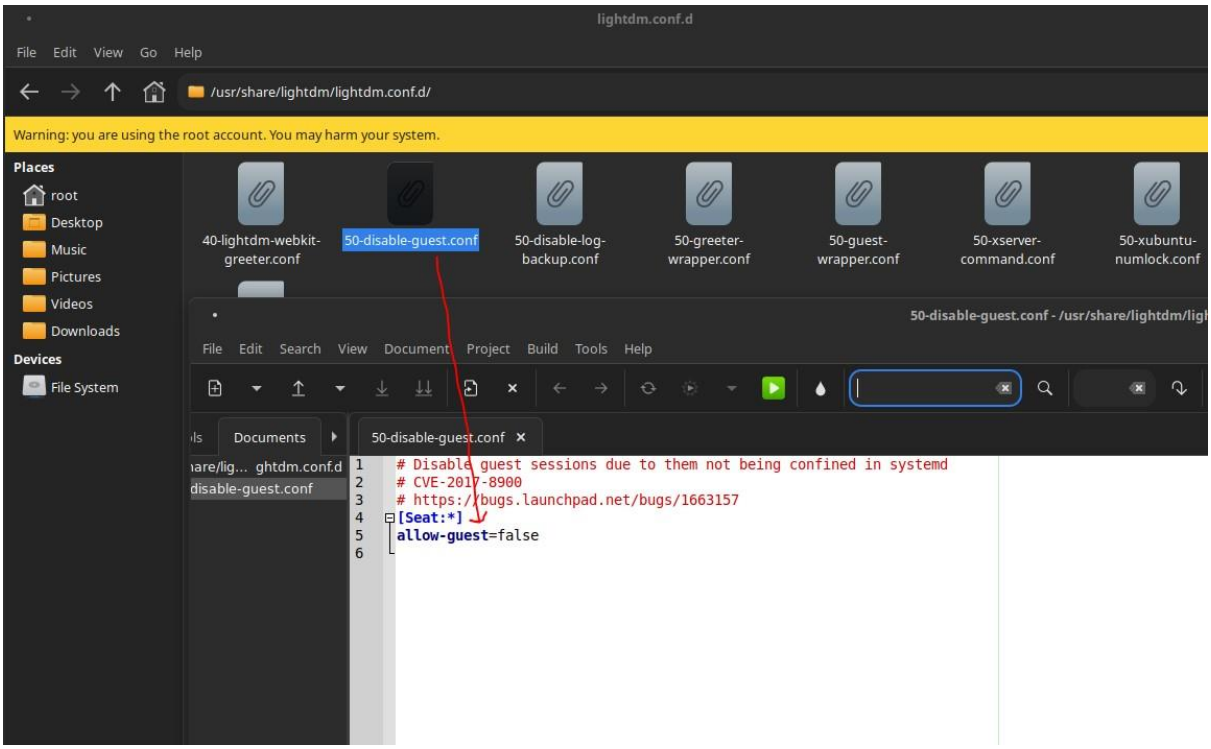
Target	Filename	Typical uses
all shells	.login_conf	Sets user-specific login defaults (FreeBSD)
sh	.profile	Sets search path, terminal type, and environment
bash [a]	.bashrc	Sets the terminal type (if needed) Sets **biff** and **mesg** switches
	.bash_profile	Sets up environment variables Sets command aliases Sets the search path Sets the **umask** value to control permissions Sets CDPATH for filename searches Sets the PS1 (prompt) and HISTCONTROL variables
csh/tcsh	.login	Read by "login" instances of **csh**
	.cshrc	Read by all instances of **csh**
vi/vim	.vimrc/.viminfo	Sets **vi**/**vim** editor options
emacs	.emacs	Sets **emacs** editor options and key bindings
git	.gitconfig	Sets user, editor, color, and alias options for Git
GNOME	.gconf	GNOME user configuration via **gconf**
	.gconfpath	Path for additional user configuration via **gconf**
KDE	.kde/	Directory of configuration files

Disabling the user feature

If you prefer to not allow guest access to your computer, you can disable the *Guest Session* feature.
To do so, press

Ctrl + Alt + T to open a terminal window, and then run this command (it's one long command, even if it may be shown wrapped on the screen - copy and paste to get it right):

```
sudo sh -c 'printf " [SeatDefaults]\nallowguest=false\n"
>/usr/share/lightdm/lightdm.conf. no-guest.conf'
```



The command creates a small configuration file. To re-enable *Guest Session*, simply remove that file:

```
sudo rm /usr/share/lightdm/lightdm.conf.d no-guest.conf
```

Security parameter configuration

PAM: Pluggable Authentication Modules

User accounts are traditionally secured by passwords stored (in encrypted form) in the **/etc/shadow** or **/etc/master.passwd** file or an equivalent network database. Many programs may need to validate accounts, including **login**, **sudo**, **su**, and any program that accepts logins on a GUI workstation.

These programs really shouldn't have hard-coded expectations about how passwords are to be encrypted or verified. Ideally, they shouldn't even assume that passwords are in use at all. What if you want to use biometric identification, a network identity system, or some kind of two-factor authentication? Pluggable Authentication Modules to the rescue!

PAM is a wrapper for a variety of method-specific authentication libraries. Administrators specify the authentication methods they want the system to use, along with the appropriate contexts for each one. Programs that require user authentication simply call the PAM system rather than implement their own forms of authentication. PAM in turn calls the authentication library specified by the system administrator.

Strictly speaking PAM is an authentication technology, not an access control technology. That is; instead of addressing the question "Does user X have permission to perform operation Y?", it helps answer the precursor question, "How do I know this is really user X?"
PAM is an important component of the access control chain on most systems, and PAM configuration is a common administrative task.

Kerberos: network cryptographic authentication

Like PAM, Kerberos deals with authentication rather than access control per se. But whereas PAM is an authentication *framework*, Kerberos is a specific authentication *method*. At sites that use Kerberos, PAM and Kerberos generally work together, PAM being the wrapper and Kerberos the actual implementation.
Kerberos uses a trusted third party (a server) to perform authentication for an entire network. You don't authenticate yourself to the machine you are using, but provide your credentials to the Kerberos service. Kerberos then issues cryptographic credentials that you can present to other services as evidence of your identity.

Choosing secure passwords

Passwords must be complex enough to not be easily guessed from e.g. personal information, or **cracked** using methods like social engineering or brute-force

Password hashes

By default, Arch stores the hashed user passwords in the root-only`/etc/shadow` file, separated from the other user parameters stored in the world-readable `/etc/passwd` file, see **Users and groups#User database**. See also **#Restricting root**.

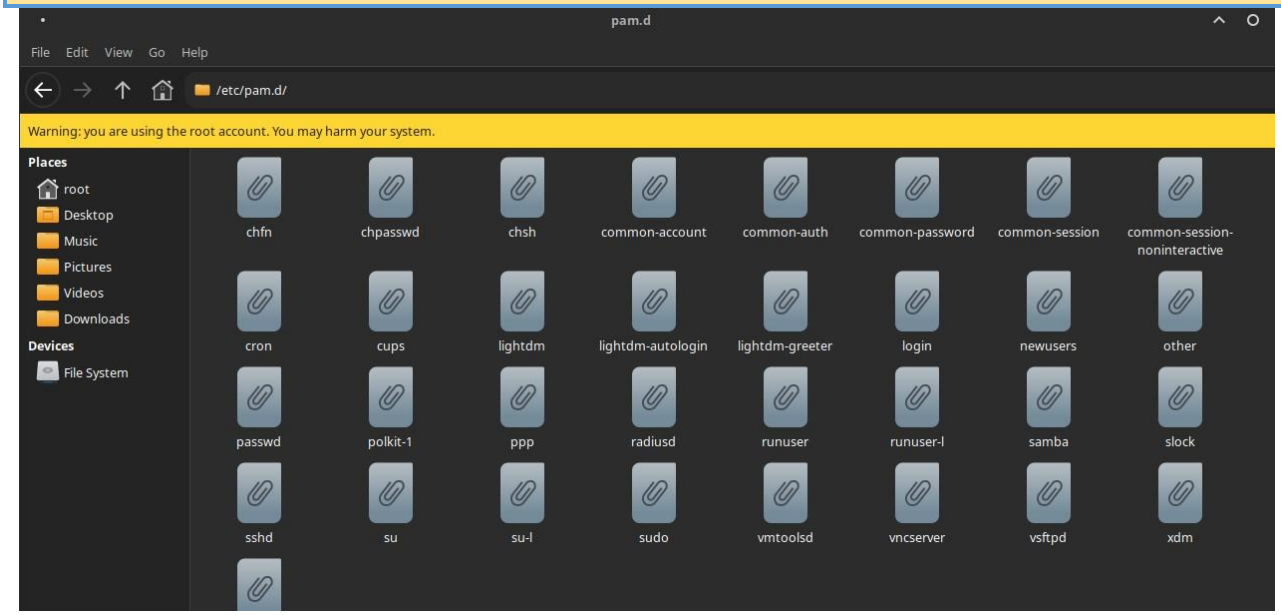
Passwords are set with the **passwd** command, which **stretches** them with the **crypt** function and then saves them in `/etc/shadow`. See also **SHA password hashes**. The passwords are also **salted** in order to defend them against **rainbow table** attacks.

Enforcing strong passwords with pam_pwquality

pam_pwquality provides protection against **Dictionary attacks** and helps configure a password policy that can be enforced throughout the system. It is based on *pam_cracklib*

Predator-OS policy is in the following path:

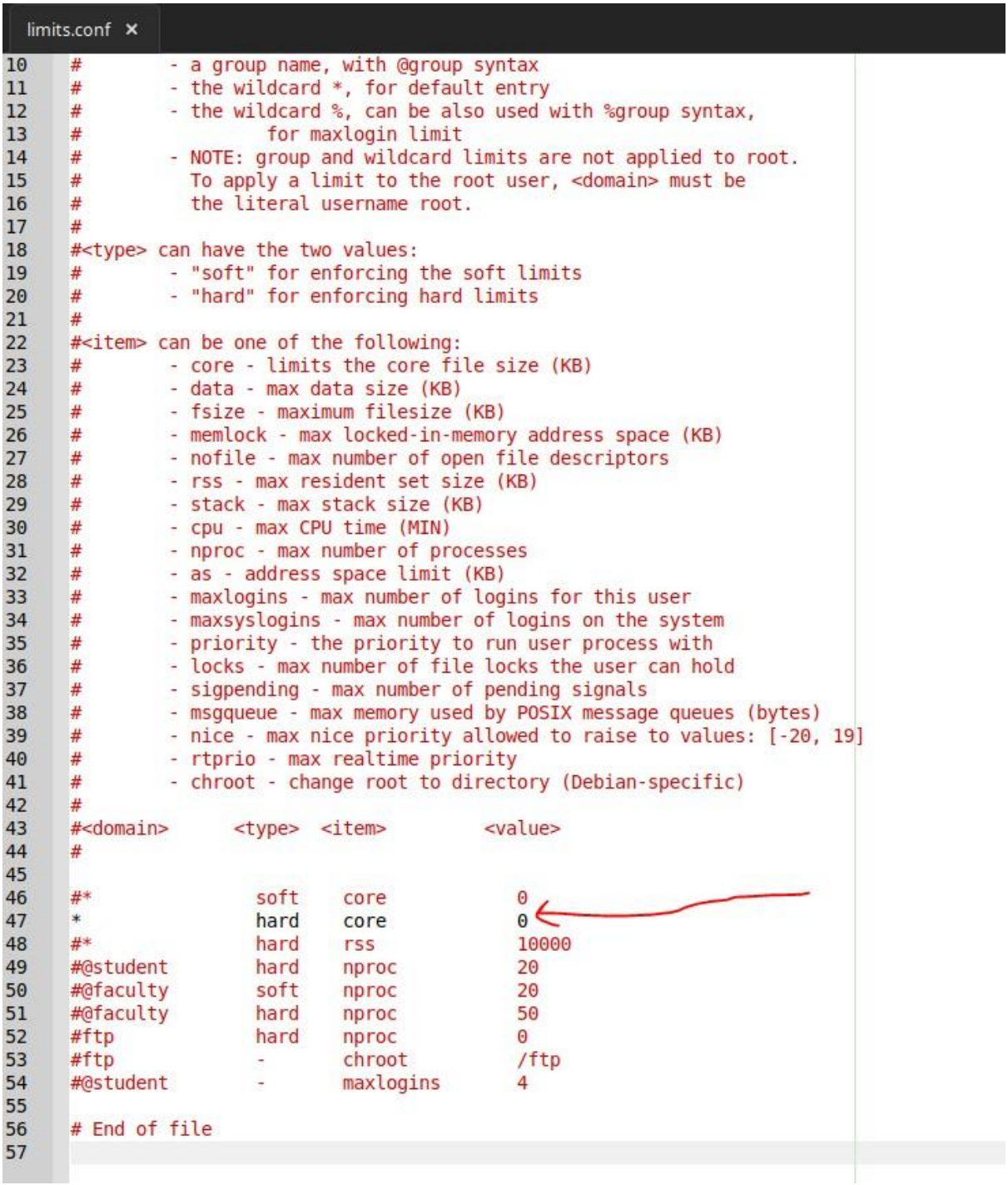
```
/etc/pam.d/passwd
```



Limit amount of processes

On systems with many, or untrusted users, it is important to limit the number of processes each can run at once, therefore preventing **fork bombs** and other denial of service attacks. `/etc/security/limits.conf` determines how many processes each user, or group can have open.

```
* soft nproc 0
* hard nproc 0
```



Restricting root login

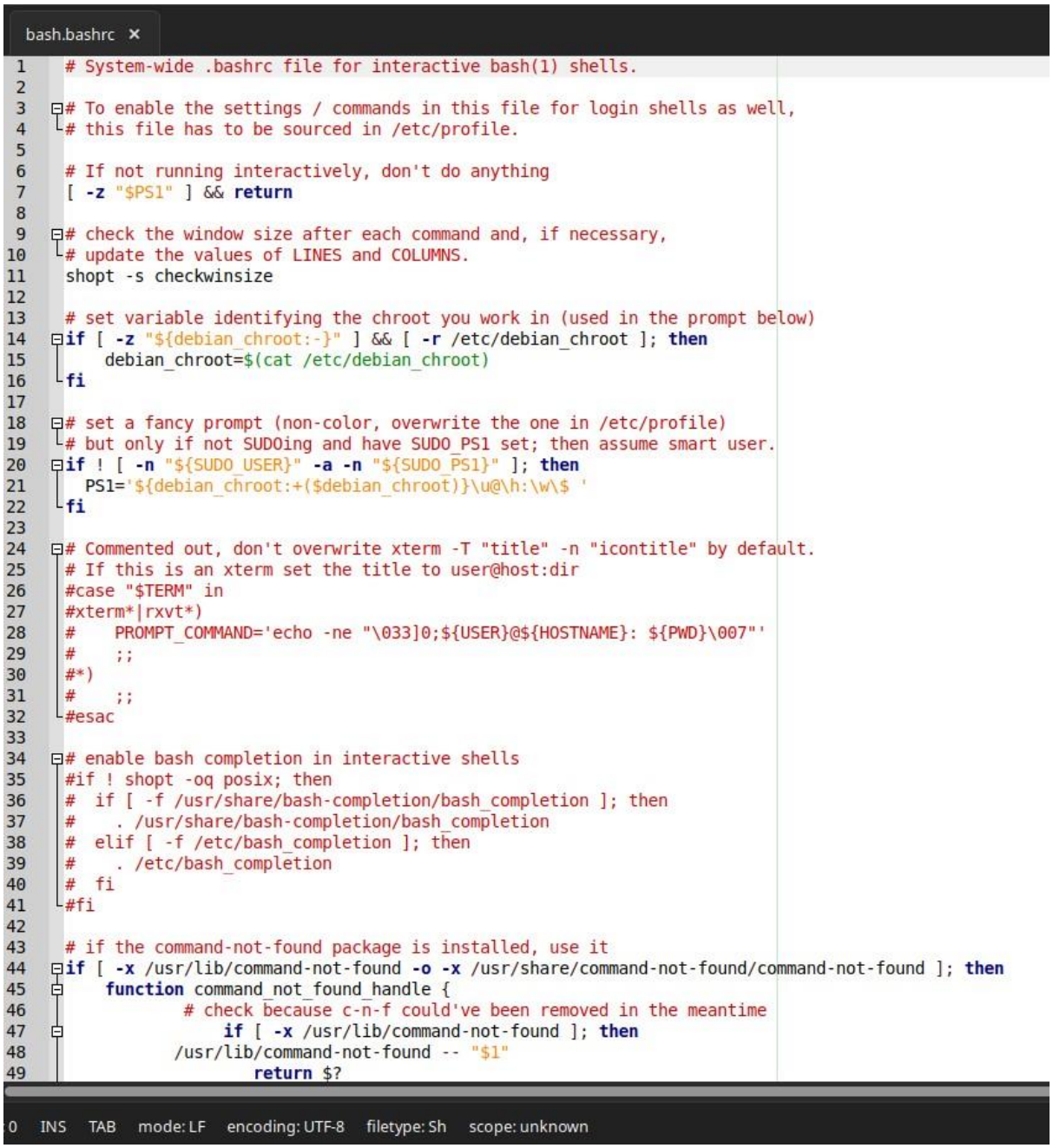
Once **sudo** is properly configured, full root access can be heavily restricted or denied without losing much usability. To disable root, but still allowing to use **sudo**, `passwd --lock root` you can use .

Shell Environment

Command interpreters (or shells) can be a user's first point of contact with the computer, and they must therefore be rather friendly. Most of them use initialization scripts that allow configuration of their behavior (automatic completion, prompt text, etc.).

bash, the standard shell, uses the **/etc/bash.bashrc** initialization script for "interactive" shells, and **/etc/profile** for "login" shells.

In simple terms, a login shell is invoked when you login to the console either locally or remotely via **ssh**, or when you run an explicit **bash -login** command. Regardless of whether it is a login shell or not, a shell can be interactive (in an **xterm**-type terminal for instance); or non-interactive (when executing a script).



For **bash**, it is useful to install and activate "automatic completion". The package bash-completion contains these completions for most common programs and is usually enabled if the user's **.bashrc** configuration file was copied from **/etc/skel/.bashrc**. Otherwise it can be enabled via **/etc/bash.bashrc** (simply uncomment a few lines) or **/etc/profile**.

Automatic completion

Many command interpreters provide a completion feature, which allows the shell to automatically complete a partially typed command name or argument when the user hits the **Tab** key. This lets users work more efficiently and be less error-prone.
Bash completion.
Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash can run most sh scripts without modification. bash-completion is a collection of shell functions that take advantage of the programmable completion feature of bash on Debian stable Linux. This page shows how to install and enable Bash auto completion in Debian stable Linux.

1. Install bash-completion package on Debian stable by running:

```
$ sudo apt install bash-completion
```

```
if [ -f /usr/share/bash-completion/bash_completion ]; then
source /usr/share/bash-completion/bash_completion
elif [ -f /etc/bash_completion ]; then
source /etc/bash_completion fi
```

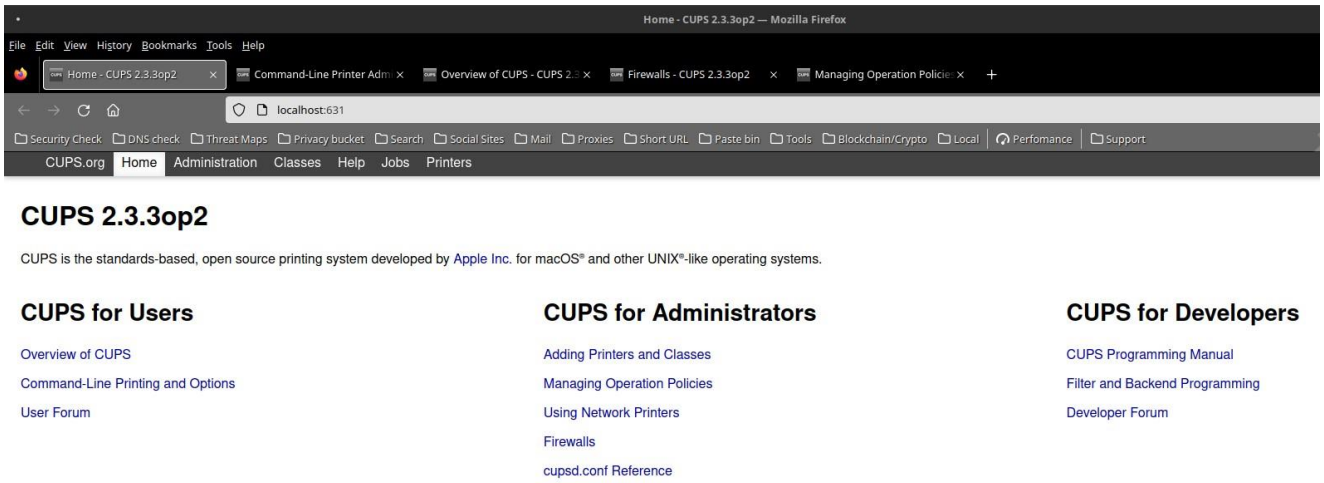
Environment variables

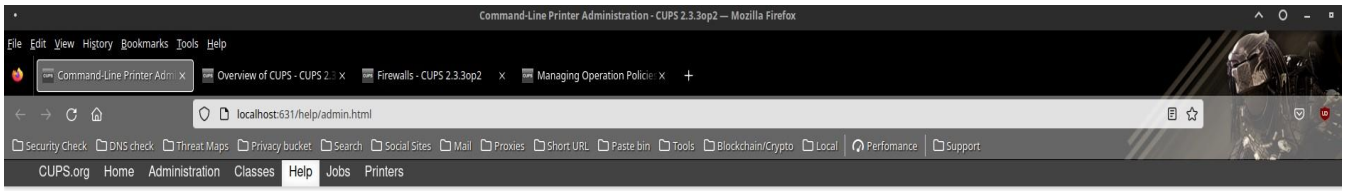
Environment variables allow storage of global settings for the shell or various other programs called. They are contextual (each process has its own set of environment variables) but inheritable. This last characteristic offers the possibility for a login shell to declare variables which will be passed down to all programs it executes. Setting default environment variables is an important element of shell configuration. Leaving aside the variables specific to a shell, it is preferable to place system wide variables in the **/etc/environment** file

Printer Configuration

Printer configuration used to cause a great many headaches for administrators and users alike. These headaches are now mostly a thing of the past, thanks to CUPS, the free print server using IPP, the Internet Printing Protocol.

The command **apt install cups** will install CUPS and the filters. It will also install the recommended printer-driver-gutenprint to provide a driver for a wide range of printers, but, unless the printer is being operated driverlessly, an alternative printerdriver might be needed for the particular device.
The printing system is administered easily through a web interface accessible at the local address **http://localhost:631/**. Members of the **lpadmin** group can add and remove USB and network printers and administer most aspects of their behavior. Similar administration tasks can also be carried out via the graphical interface provided by a desktop environment or the **system-config-printer** graphical interface.





Command-Line Printer Administration

Search in Command-Line Printer Administration: Search Clear

View Printable Version

This help document describes how to configure and manage destinations with CUPS.

Introduction

Destinations are individual printers and classes (pools) of printers. Printers use a description file with one or more driver ("filter") programs that communicate with the printer through a "backend" program. CUPS currently uses PPD (PostScript Printer Description) files to describe the printer and driver programs needed, some of which come with CUPS while others come with your operating system or Linux distribution. Backends are specified using a URI (Universal Resource Identifier) where the URI scheme is the backend name, e.g., "ipp://11.22.33.44/ipp/print" specifies the "ipp" backend - like PPD files, some backends come with CUPS while others come with your operating system.

Classes are associated with one or more printers and are typically used to distribute print jobs amongst a group of printers or provide redundancy or high availability when printing. Print jobs sent to a class are forwarded to the next available printer in the class.

The `lpadmin(8)` program is used to add, modify, or delete destinations, while the `lpinfo(8)` command is used to list the available printer drivers and backends. The `cupsctl(8)` program is used to manage the printing system as a whole, including things like debug logging and printer sharing. The CUPS web interface ("http://localhost:631" or "https://servername:631") can also be used, and most operating systems provide their own GUI administration tools.

Managing Printers

The `lpadmin` command is used to create, modify, or delete a printer. The `-p` option specifies a printer to create or modify:

```
lpadmin -p printername ...
```

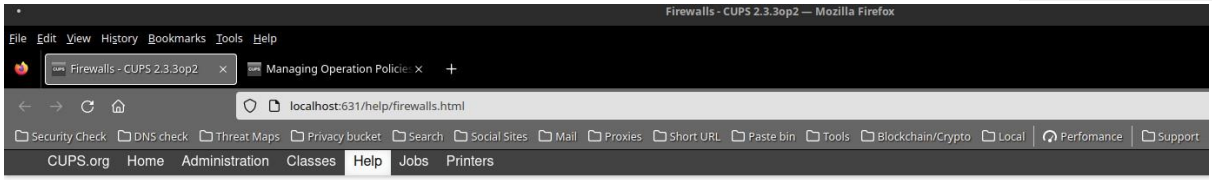
The `lpadmin` accepts several additional options after `-p printername` when adding or modifying a printer:

`-D "description"`
Sets the description of the printer which is often shown instead of the printer name, for example "HP LaserJet".

`-E`
Enables the printer and accepts new print jobs.

Online Help Documents

All Documents



Firewalls

Search in Firewalls: Search Clear

View Printable Version

This help document describes the ports that CUPS uses so that firewall administrators can allow traffic used for printing.

Ports Used for Printer Sharing

Table 1 lists the ports that are used for IPP printer sharing via CUPS.

Table 1: Ports Used for IPP Printer Sharing

(Destination) Port	TCP/UDP	Direction	Description
53 (DNS)	TCP/UDP	OUT	Domain Name System lookups and service registrations.
631 (IPP/IPPS)	TCP	IN	Internet Printing Protocol requests and responses (print jobs, status monitoring, etc.)
5353 (mDNS)	UDP	IN+OUT	Multicast DNS lookups and service registrations.

Table 2 lists the ports that are used for SMB (Windows) printer sharing, typically via the Samba software.

Table 2: Ports Used for SMB Printer Sharing

(Destination) Port(s)	TCP/UDP	Direction	Description
137 (WINS)	UDP	IN+OUT	Windows Internet Naming Service (name lookup for SMB printing).
139 (SMB)	TCP	IN	Windows SMB printing.
445 (SMBDS)	TCP	IN+OUT	Windows SMB Domain Server (authenticated SMB printing).

Ports Used for Network Printers

Table 3 lists the ports for outgoing network traffic that are used for network printers.

Bootloader

It is probably already functional, but it is always good to know how to configure and install the bootloader in case it disappears from the Master Boot Record. This can occur after installation of another operating system, such as Windows. The following information can also help you to modify the bootloader configuration if needed.

Legacy BIOS Traditional BIOS assumes that the boot device starts with a record called the MBR (Master Boot Record). The MBR includes both a first-stage boot loader (aka "boot block") and a primitive disk partitioning table. The amount of space available for the boot loader is so small (less than 512 bytes) that it's not able to do much other than load and run a second-stage boot loader. Neither the boot block nor the BIOS is sophisticated enough to read any type of standard filesystem, so the second-stage boot loader must be kept somewhere easy to find. In one typical scenario, the boot block reads the partitioning information from the MBR and identifies the disk partition marked as "active." It then reads and executes the second-stage boot loader from the beginning of that partition. This scheme is known as a volume boot record. Alternatively, the second-stage boot loader can live in the dead zone that lies between the MBR and the beginning of the first disk partition. For historical reasons, the first partition does not start until the 64th disk block, so this zone normally contains at least 32KB of storage: still not a lot, but enough to store a filesystem driver. This storage scheme is commonly used by the GRUB boot loader; see page 35. To effect a successful boot, all components of the boot chain must be properly installed and compatible with one another. The MBR boot block is OS-agnostic, but because it assumes a particular location for the second stage, there may be multiple versions that can be installed. The second-stage loader is generally knowledgeable about operating systems and filesystems (it may support several of each), and usually has configuration options of its own

UEFI The UEFI specification includes a modern disk partitioning scheme known as GPT (GUID Partition Table, where GUID stands for "globally unique identifier"). UEFI also understands FAT (File Allocation Table) filesystems, a simple but functional layout that originated in MS-DOS. These features combine to define the concept of an EFI System Partition (ESP). At boot time, the firmware consults the GPT partition table to identify the ESP. It then reads the configured target application directly from a file in the ESP and executes it.

Because the ESP is just a generic FAT filesystem, it can be mounted, read, written, and maintained by any operating system. No "mystery meat" boot blocks are required anywhere on the disk.3 In fact, no boot loader at all is technically required. The UEFI boot target can be a UNIX or Linux kernel that has been configured for direct UEFI loading, thus effecting a loader-less bootstrap. In practice, though, most systems still use a boot loader, partly because that makes it easier to maintain compatibility with legacy BIOSes. UEFI saves the pathname to load from the ESP as a configuration parameter. With no configuration, it looks for a standard path, usually /efi/boot/bootx64.efi on modern Intel systems. A more typical path on a configured system (this one for Debian stable and the GRUB boot loader) would be /efi/Debian stable/grubx64.efi. Other distributions follow a similar convention.

Because UEFI has a formal API, you can examine and modify UEFI variables (including boot menu entries) on a running system. For example, efibootmgr -v shows the following summary of the boot configuration:

```
$ efibootmgr -v
```

GRUB Configuration

GRUB (GRand Unified Bootloader) is more recent. It is not necessary to invoke it after each update of the kernel; GRUB knows how to read the filesystems and find the position of the kernel on the disk by itself. To install it on the MBR of the first disk, simply type **grub-install /dev/sda**. This will overwrite the MBR, so be careful not to overwrite the wrong location. While it is also possible to install GRUB into a partition boot record, beware that it is usually a mistake and doing **grub-install /dev/sda1** has not the same meaning as grub-install /dev/sda.

GRUB 2 configuration is stored in /boot/grub/grub.cfg,

but this file is generated from others. Be careful not to modify it by hand, since such local modifications will be lost the next time **update-grub** is run (which may occur upon update of various packages). The most common modifications of the **/boot/grub/grub.cfg** file (to add command line parameters to the kernel or change the duration that the menu is displayed, for example) are made through the variables in **/etc/default/grub**. To add entries to the menu, you can either create a **/boot/grub/custom.cfg** file or modify the **/etc/grub.d/40_custom** file. For more complex configurations, you can modify other files in **/etc/grub.d**, or add to them; these scripts should return configuration snippets, possibly by making use of

external programs. These scripts are the ones that will update the list of kernels to boot: **10_linux** takes into consideration the installed Linux kernels; **20_linux_xen** takes into account Xen virtual systems, and **30_osprober** adds other existing operating systems (Windows, OS X, Hurd), kernel images, and BIOS/EFI access options to the menu.

The config file is called grub.cfg, and it's usually kept in /boot/grub (/boot/grub2 in Red Hat and CentOS) along with a selection of other resources and code modules that GRUB might need to access at boot time.5 Changing the boot configuration is a simple matter of updating the grub.cfg file. Although you can create the grub.cfg file yourself, it's more common to generate it with the grub-mkconfig utility, which is called grub2-mkconfig on Red Hat and CentOS and wrapped as update-grub on Debian and Debian stable. In fact, most distributions assume that grub.cfg can be regenerated at will, and they do so automatically after updates. If you don't take steps to prevent this, your handcrafted grub.cfg file will get clobbered. As with all things Linux, distributions configure grub-mkconfig in a variety of ways. Most commonly, the configuration is specified in /etc/default/grub in the form of sh variable assignments

Common GRUB configuration options from /etc/default/grub Shell variable name Contents or function GRUB_BACKGROUND Background image a GRUB_CMDLINE_LINUX Kernel parameters to add to menu entries for Linux b GRUB_DEFAULT Number or title of the default menu entry GRUB_DISABLE_RECOVERY Prevents the generation of recovery mode entries GRUB_PRELOAD_MODULES List of GRUB modules to be loaded as early as possible GRUB_TIMEOUT Seconds to display the boot menu before autoboot.

```
# If you change this file, run 'update-grub' afterwards to update #
/boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT="0"
GRUB_TIMEOUT_STYLE="menu"
GRUB_TIMEOUT="15"
GRUB_DISTRIBUTOR="`lsb_release -i -s 2> /dev/null || echo Debian`"
GRUB_CMDLINE_LINUX_DEFAULT="mitigations=off loglevel=0 nowatchdog
intel_pstate=false quiet splash"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt
priority=critical"
```

```
GRUB_DISABLE_OS_PROBER="false"
 # Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"
 # Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL="console"

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
GRUB_GFXMODE="1024x768x24"
 # Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID="true"

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
GRUB_INIT_TUNE="480 440 1"
 #GRUB_HIDDEN_TIMEOUT="0"
 GRUB_SAVEDEFAULT="false" export GRUB_COLOR_NORMAL="white/black"
export GRUB_COLOR_HIGHLIGHT="yellow/black"
export GRUB_MENU_PICTURE="/usr/share/backgrounds/grub.PNG"
```

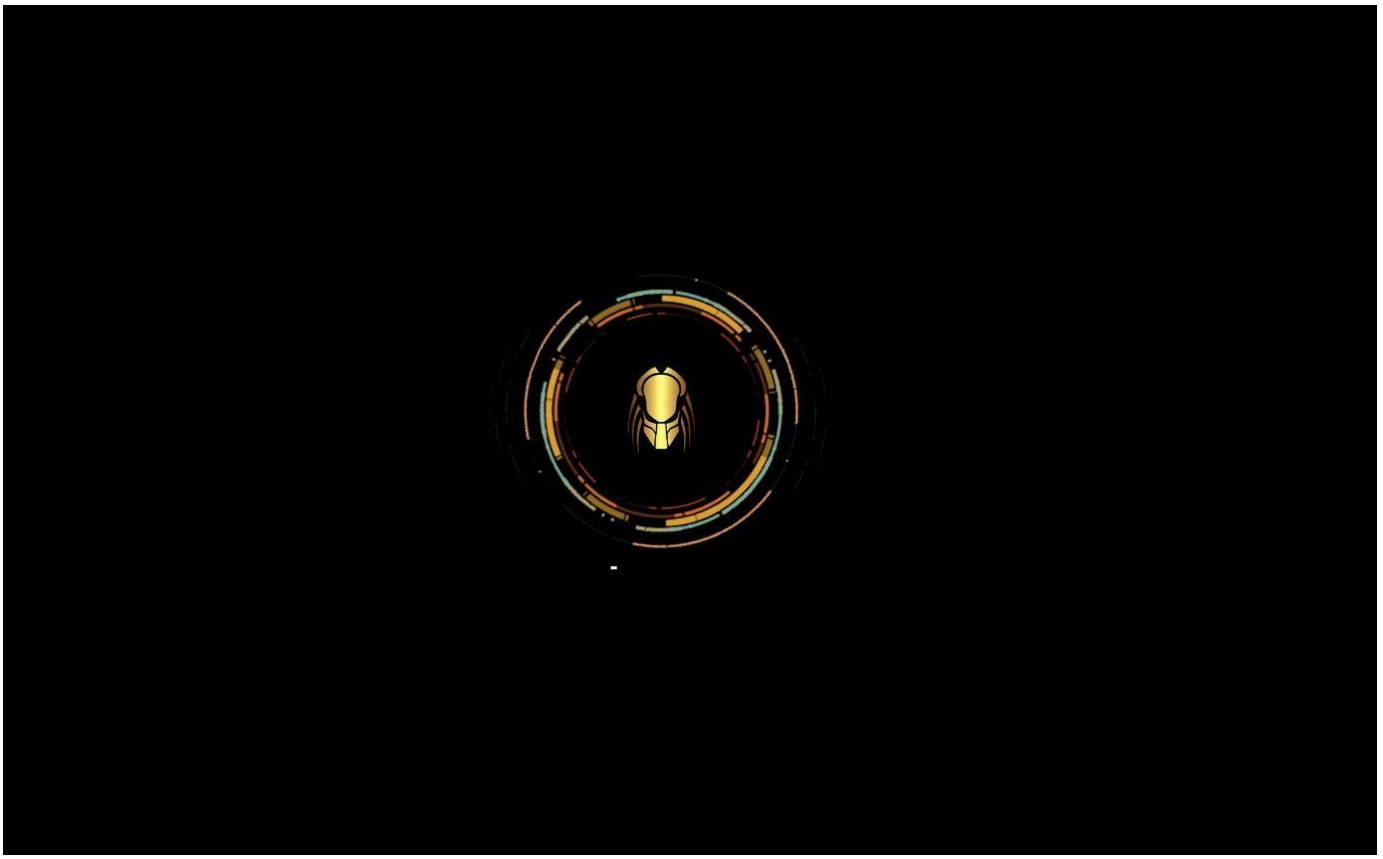
Common GRUB configuration options from /etc/default/grub

Shell variable name	Contents or function
GRUB_BACKGROUND	Background image [a]
GRUB_CMDLINE_LINUX	Kernel parameters to add to menu entries for Linux [b]
GRUB_DEFAULT	Number or title of the default menu entry
GRUB_DISABLE_RECOVERY	Prevents the generation of recovery mode entries
GRUB_PRELOAD_MODULES	List of GRUB modules to be loaded as early as possible
GRUB_TIMEOUT	Seconds to display the boot menu before autoboot

The background image must be a **.png**, **.tga**, **.jpg**, or **.jpeg** file.

After editing /etc/default/grub, run update-grub or grub2-mkconfig to translate your configuration into a proper grub.cfg file


predator-os CD
Boot in Live/install Mode
Boot in Direct Install
Boot in Full RAM Mode
Boot in Forensics Mode
Boot in Persistent Mode
Boot in persistence Encrypted Mode
Boot in safe graphics Mode
Boot in text Mode
Boot in No Acpi Mode
Boot in iommu Mode
Boot in Acpi off Mode
Boot in Nomodeset Mode
Press [Tab] to edit options
PREDATOR-OS



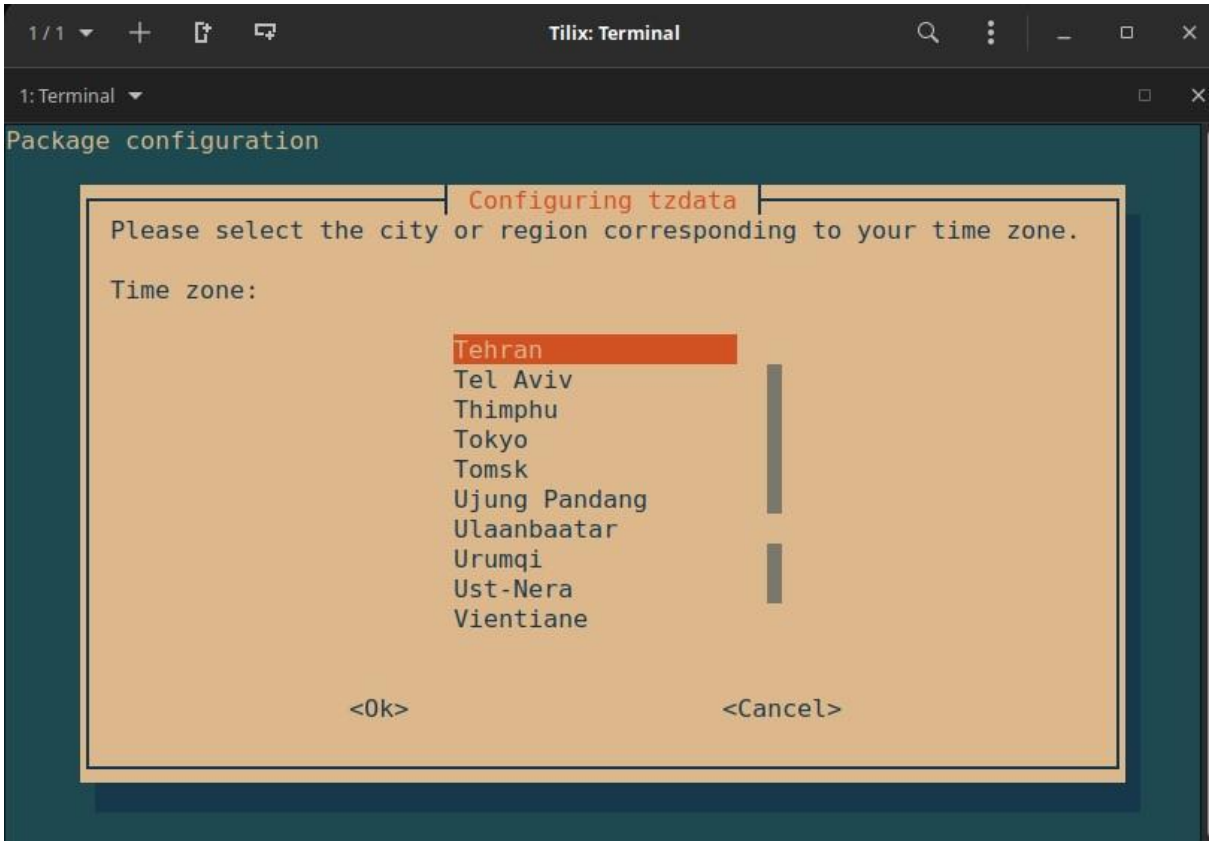
```
[  OK  ] Reached target Local Encrypted Volumes.
[  OK  ] Reached target System Initialization.
[  OK  ] Started ACPI Events Check.
[  OK  ] Started resolvconf-pull-resolved.path.
[  OK  ] Started Trigger anacron every hour.
[  OK  ] Started Periodic ext4 Online Metadata Check for All Filesystems.
[  OK  ] Started Discard unused blocks once a week.
[  OK  ] Started Weekly GeoIP update.
[  OK  ] Started Daily timer for the Lynis security audit and vulnerability scanner.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Timers.
[  OK  ] Listening on ACPI fakekey daemon FIFO.
[  OK  ] Listening on ACPID Listen Socket.
[  OK  ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Listening on GPS (Global Positioning System) Daemon Sockets.
[  OK  ] Listening on PC/SC Smart Card Daemon Activation Socket.
[  OK  ] Listening on UUID daemon activation socket.
[  OK  ] Reached target Sockets.
[  OK  ] Reached target Basic System.
         Starting LSB: Run firstboot items fo… remastered system has been installed...
         Starting Accounts Service...
[  OK  ] Started ACPI fakekey daemon.
acpi-fakekey.service
[  OK  ] Started ACPI event daemon.
acpid.service
         Starting arpwatch service...
         Starting Avahi mDNS/DNS-SD Stack...
[  OK  ] Started Regular background program processing daemon.
         Starting cryptmount startup...
cron.service
[  OK  ] Started D-Bus System Message Bus.
dbus.service
         Starting Network Manager...
         Starting Remove Stale Online ext4 Metadata Check Snapshots...
```

Using GRUB with EFI and Secure Boot

Using GRUB to boot either a traditional BIOS system (legacy or UEFI-CSM) or a UEFI system is quite different. Fortunately, the user does not need to know the differences because linux provides different packages for each purpose and the installer automatically cares about which one(s) to choose. The grub-pc package is chosen for legacy systems, where GRUB is installed into the MBR, while UEFI systems require grub-efi-arch, where GRUB is installed into the EFI System Partition (ESP). The latter requires a GTP partition table as well as an EFI partition.

o switch an existing system (supporting UEFI) from legacy to UEFI boot mode not only requires to switch the GRUB packages on the system, but also to adjust the partition table and the to create an EFI partition (probably including resizing existing partitions to create the necessary free space). It is therefore quite an elaborate process and we cannot cover it here. Fortunately, there are some manuals by bloggers describing the necessary procedures.

The timezone, configured during initial installation, is a configuration item for the tzdata package. To modify it, use the **dpkg-reconfigure tzdata** command, which allows you to choose the timezone to be used in an interactive manner. Its configuration is stored in the **/etc/timezone** file. Additionally, **/etc/localtime** becomes a symbolic link to the corresponding file in the **/usr/share/zoneinfo**; the file that contains the rules governing the dates where daylight saving time (DST) is active, for countries that use it.



When you need to temporarily change the timezone, use the TZ environment variable, which takes priority over the configured system default:

```
$ date
```

Thu Sep 2 22:29:48 CEST 2023

```
$ TZ="Pacific/Honolulu" date
```

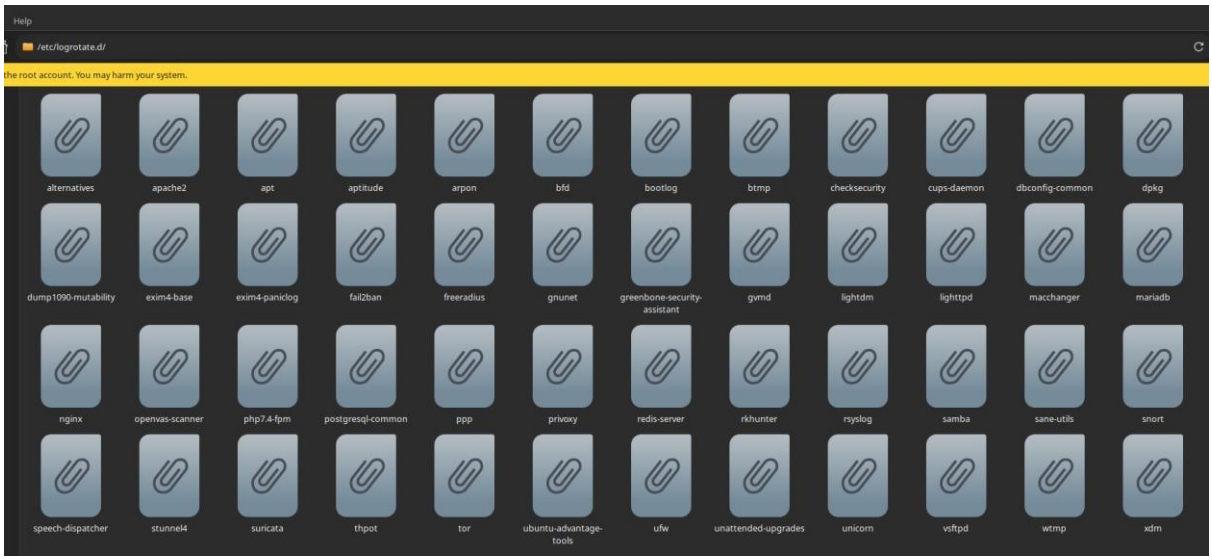
Thu 02 Sep 2023 10:31:01 AM HST

For Workstations

Since work stations are regularly rebooted (even if only to save energy), synchronizing them by NTP at boot is enough. To do so, simply install the ntpdate package. You can change the NTP server used if needed by modifying the **/etc/default/ntpdate** file.

Rotating Log Files

Log files can grow, fast, and it is necessary to archive them. The most common scheme is a rotating archive: the log file is regularly archived, and only the latest X archives are retained. **logrotate**, the program responsible for these rotations, follows directives given in the **/etc/logrotate.conf** file and all of the files in the **/etc/logrotate.d/** directory. The administrator may modify these files, if they wish to adapt the log rotation policy. The logrotate(1) man page describes all of the options available in these configuration files. You may want to increase the number of files retained in log rotation, or move the log files to a specific directory dedicated to archiving them rather than delete them. You could also send them by e-mail to archive them elsewhere.



/etc/logrotate.d/ directory

Source of : /etc/logrotate.conf

```
# see “man logrotate” for details

# global options do not affect preceding include directives

# rotate log files weekly
```

```
weekly

# keep 4 weeks worth of backlogs rotate
4

# create new (empty) log files after rotating old ones create

# use date as a suffix of the rotated file
#dateext

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# system-specific logs may also be configured here.
```

System daemons, the kernel, and custom applications all emit operational data that is logged and eventually ends up on your finite sized disks. This data has a limited useful life and may need to be summarized, filtered, searched, analyzed, compressed, and archived before it is eventually discarded. Access and audit logs may need to be managed closely according to regulatory retention rules or site security policies.

A log message is usually a line of text with a few properties attached, including a time stamp, the type and severity of the event, and a process name and ID (PID). The message itself can range from an innocuous note about a new process starting up to a critical error condition or stack trace. It's the responsibility of system administrators to glean useful, actionable information from this ongoing torrent of messages. This task is known generically as log management, and it can be divided into a few major subtasks:

- Collecting logs from a variety of sources
- Providing a structured interface for querying, analyzing, filtering, and monitoring messages
- Managing the retention and expiration of messages so that information is kept as long as it is potentially useful or legally required, but not indefinitely

UNIX has historically managed logs through an integrated but somewhat rudimentary system, known as syslog, that presents applications with a standardized

interface for submitting log messages. Syslog sorts messages and saves them to files or forwards them to another host over the network. Unfortunately, syslog tackles only the first of the logging chores listed above (message collection), and its stock configuration differs widely among operating systems.

Perhaps because of syslog's shortcomings, many applications, network daemons, startup scripts, and other logging vigilantes bypass syslog entirely and write to their own ad hoc log files. This lawlessness has resulted in a complement of logs that varies significantly among flavors of UNIX and even among Linux distributions. Linux's systemd journal represents a second attempt to bring sanity to the logging madness. The journal collects messages, stores them in an indexed and compressed binary format, and furnishes a command-line interface for viewing and filtering logs. The journal can stand alone, or it can coexist with the syslog daemon with varying degrees of integration, depending on the configuration.

A variety of third party tools (both proprietary and open source) address the more complex problem of curating messages that originate from a large network of systems. These tools feature such aids as graphical interfaces, query languages, data visualization, alerting, and automated anomaly detection. They can scale to handle message volumes on the order of terabytes per day. You can subscribe to these products as a cloud service or host them yourself on a private network.

Exhibit A on the next page depicts the architecture of a site that uses all the log management services mentioned above. Administrators and other interested parties can run a GUI against the centralized log cluster to review log messages from systems across the network. Administrators can also log in to individual nodes and access messages through the systemd journal or the plain text files written by syslog. When debugging problems and errors, experienced administrators turn to the logs sooner rather than later. Log files often contain important hints that point toward the source of vexing configuration errors, software bugs, and security issues. Logs are the first place you should look when a daemon crashes or refuses to start, or when a chronic error plagues a system that is trying to boot.

The importance of having a well-defined, site-wide logging strategy has grown along with the adoption of formal IT standards such as PCI DSS, COBIT, and ISO 27001, as well as with the maturing of regulations for individual industries. Today, these external standards may require you to maintain a centralized, hardened, enterprise-wide repository for log activity, with time stamps validated by NTP and with a strictly defined retention schedule.1 However, even sites without regulatory or compliance requirements can benefit from centralized logging.

Log locations

UNIX is often criticized for being inconsistent, and indeed it is. Just take a look at a directory of log files and you're sure to find some with names like maillog, some like cron.log, and some that use various distribution- and daemon-specific naming conventions. By default, most of these files are found in /var/log, but some renegade applications write their log files elsewhere on the filesystem.

Table 10.1 compiles information about some of the more common log files on our example systems. The table lists the following:

- The log files to archive, summarize, or truncate
- The program that creates each
- An indication of how each filename is specified
- The frequency of cleanup that we consider reasonable
- The systems (among our examples) that use the log file
- A description of the file's contents

The systemd journal

In accordance with its mission to replace all other Linux subsystems, **systemd** includes a logging daemon called **systemd-journald**. It duplicates most of syslog's functions but can also run peacefully in tandem with syslog, depending on how you or the system have configured it. If you're leery of switching to **systemd** because syslog has always "just worked" for you, spend some time to get to know **systemd**. After a little practice, you may be pleasantly surprised.

Unlike syslog, which typically saves log messages to plain text files, the **systemd** journal stores messages in a binary format. All message attributes are indexed automatically, which makes the log easier and faster to search. As discussed above, you can use the **journalctl** command to review messages stored in the journal. The journal collects and indexes messages from several sources:

- The **/dev/log** socket, to harvest messages from software that submits messages according to syslog conventions
- The device file **/dev/kmsg**, to collect messages from the Linux kernel. The **systemd** journal daemon replaces the traditional **klogd** process that previously listened on this channel and formerly forwarded the kernel messages to syslog.
- The UNIX socket **/run/systemd/journal/stdout**, to service software that writes log messages to standard output
- The UNIX socket **/run/systemd/journal/socket**, to service software that submits messages through the **systemd** journal API
- Audit messages from the kernel's **auditd** daemon

Intrepid administrators can use the **systemd-journal-remote** utility (and its relatives, **systemd-journal-gateway** and **systemd-journal-upload**,) to stream serialized journal messages over the network to a remote journal. Unfortunately, this feature does not come preinstalled on vanilla distributions. As of this writing, packages are available for Debian and Debian stable but not for Red Hat or CentOS. We expect this lapse to be rectified soon; in the meantime, we recommend sticking with syslog if you need to forward log messages among systems.

Configuring the systemd journal

The default journal configuration file is **/etc/systemd/journald.conf**; however, this file is not intended to be edited directly. Instead, add your customized configurations to the **/etc/systemd/journald.conf.d** directory. Any files placed there with a **.conf** extension are automatically incorporated into the configuration. To set your own options, create a new **.conf** file in this directory and include the options you want. The default **journald.conf** includes a commented-out version of every possible option, along with each option's default value, so you can see at a glance which options are available. They include the maximum size of journal, the retention period for messages, and various rate-limiting settings.

/etc/systemd/journald.conf

```
#  This file is part of systemd.
#  systemd is free software; you can redistribute it and/or modify it
# Entries in this file show the compile time defaults.
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file. #
See journald.conf(5) for details.
[Journal]
Storage=none
Compress=no
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitIntervalSec=30s
#RateLimitBurst=10000
#SystemMaxUse=
```

```
#SystemKeepFree=
#SystemMaxFileSize=
#SystemMaxFiles=100
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#RuntimeMaxFiles=100
#MaxRetentionSec=
#MaxFileSec=1month
#ForwardToSyslog=yes
#ForwardToKMsg=no
#ForwardToConsole=no #ForwardToWall=yes
#TTYPath=/dev/console
#MaxLevelStore=debug
#MaxLevelSyslog=debug
#MaxLevelKMsg=notice
#MaxLevelConsole=info
#MaxLevelWall=emerg
#LineMax=48K
#ReadKMsg=yes
Audit=no
```

Journal logs help you debug your system. But for most of the time journaling may write a lot to your storage, and overtime the logs becomes huge. It's then truncated, but if that's not what you want, you can disable it by editing

```
/etc/systemd/journald.conf, and set:
Storage=none
```

systemd logging

Capturing the log messages produced by the kernel has always been something of a challenge. It became even more important with the advent of virtual and cloudbased systems, since it isn't possible to simply stand in front of these systems' consoles and watch what happens. Frequently, crucial diagnostic information was lost to the ether.

systemd alleviates this problem with a universal logging framework that includes all kernel and service messages from early boot to final shutdown. This facility, called the journal, is managed by the **journald** daemon.
System messages captured by **journald** are stored in the /**run** directory. **rsyslog** can process these messages and store them in traditional log files or forward them to a remote syslog server. You can also access the logs directly with the **journalctl** command.
Without arguments, **journalctl** displays all log entries (oldest first): $
journalctl

You can configure **journald** to retain messages from prior boots. To do this, edit /etc/systemd/journald.conf and configure the Storage attribute:
[Journal]
Storage=persistent
Once you've configured journald, you can obtain a list of prior boots with $
journalctl --list-boots

Log files locations

There are many different log files that all serve different purposes. When trying to find a log about something, you should start by identifying the most relevant file. Below is a list of common log file locations.

System logs

System logs deal with exactly that - the Debian stable system - as opposed to extra applications added by the user. These logs may contain information about authorizations, system daemons and system messages.

Authorization log

Location: /var/log/auth.log

Keeps track of authorization systems, such as password prompts, the sudo command and remote logins.

Daemon Log

Location: /var/log/daemon.log

Daemons are programs that run in the background, usually without user interaction. For example, display server, SSH sessions, printing services, bluetooth, and more.

Debug log

Location: /var/log/debug

Provides debugging information from the Debian stable system and applications.

Kernel log

Location: /var/log/kern.log

Logs from the Linux kernel.

System log

Location: /var/log/syslog

Contains more information about your system. If you can't find anything in the other logs, it's probably here.

Application logs

Some applications also create logs in /var/log. Below are some examples.

Apache logs

Location: /var/log/apache2/ (subdirectory)

Apache creates several log files in the /var/log/apache2/ subdirectory. The access.log file records all requests made to the server to access files. error.log records all errors thrown by the server.

X11 server logs

Location: /var/log/Xorg.0.log

The X11 server creates a seperate log file for each of your displays. Display numbers start at zero, so your first display (display 0) will log to Xorg.0.log. The next display (display 1) would log to Xorg.1.log, and so on.

Non-human-readable logs

Not all log files are designed to be read by humans. Some were made to be parsed by applications. Below are some of examples.

Login failures log

Location: /var/log/faillog

Contains info about login failures. You can view it with the faillog command.

Last logins log

Location: /var/log/lastlog

Contains info about last logins. You can view it with the lastlog command.

Login records log

Location: /var/log/wtmp

Syslog severity levels (descending severity)

Level	Approximate meaning
emerg	Panic situations; system is unusable
alert	Urgent situations; immediate action required
crit	Critical conditions
err	Other error conditions
warning	Warning messages
notice	Things that might merit investigation
info	Informational messages
debug	For debugging only

loglevel — Set the default console log level.

Making sense out of logs is not an easy task. Log management solutions gather and accept data from multiple sources. Those sources can have different log events structures, providing a different granularity. They may not follow common logging best practices and be hard to get some meaning from.

Because of that, it is important to follow good practices when we develop an application. One of those is keeping meaningful log levels. That allows a person who will read the logs and try to give them meaning to understand the importance of the message that they see in the text files or one of those awesome observability tools out there.

What Is a Logging Level?

A log level or log severity is a piece of information telling how important a given log message is. It is a simple, yet very powerful way of distinguishing log events from each other. If the log levels are used properly in your application, all you need

is to look at the severity first. It will tell you if you can continue sleeping during the on-call night or you need to jump out of bed right away and hit another personal best in running between your bedroom and laptop in the living room.

You can think of the log levels as a way to filter the critical information about your system state and the one that is purely informative. The log levels can help to reduce the information noise and alert fatigue.

The History of Log Levels

Before continuing with the description of the log levels themselves it would be good to know where the log levels come from. It all started with syslog. In the 80s, the Sendmail a mailer daemon project developed by Eric Allman required a logging solution. This is how Syslog was born. It was rapidly adopted by other applications in the Unix-like ecosystem and became a standard. Btw – at Sematext we do support Syslog format with Sematext Logs, our log management tool.

The console log level can also be changed by the *klogd* program, or by writing the specified level to the */proc/sys/kernel/printk* file.

The kernel log levels are:

0 (KERN_EMERG)

The system is unusable.

1 (KERN_ALERT)

Actions that must be taken care of immediately.

2 (KERN_CRIT)

Critical conditions.

3 (KERN_ERR)

Non-critical error conditions.

4 (KERN_WARNING)

Warning conditions that should be taken care of.

5 (KERN_NOTICE)

Normal, but significant events.

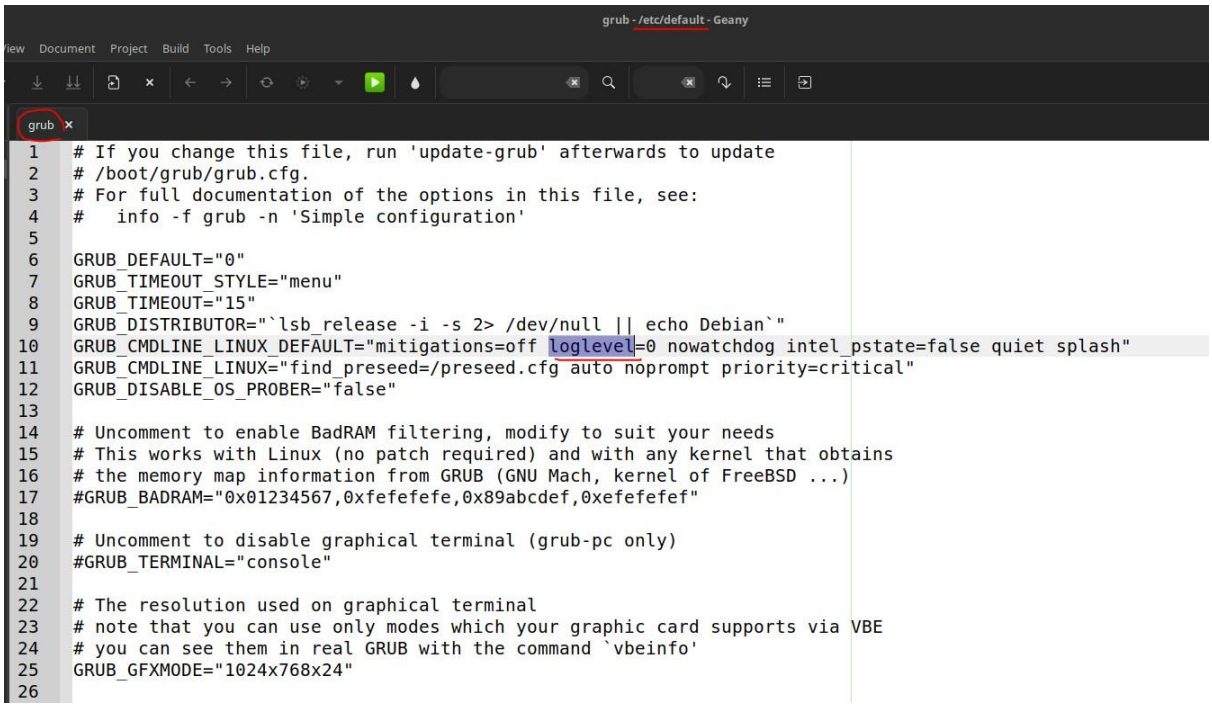
6 (KERN_INFO)

Informational messages that require no action.

7 (KERN_DEBUG)

Kernel debugging messages, output by the kernel if the developer enabled debugging at compile time.

By default, the log level of Predator-OS is 0.



Message logging with printk

printk() is one of the most widely known functions in the Linux kernel. It's the standard tool we have for printing messages and usually the most basic way of tracing and debugging. If you're familiar with printf(3) you can tell **printk()** is based on it, although it has some functional differences:

- **printk()** messages can specify a log level.
- the format string, while largely compatible with C99, does not follow the exact same specification. It has some extensions and a few limitations (no `%n` or floating point conversion specifiers). See How to get printk format specifiers right.

where `KERN_INFO` is the log level (note that it's concatenated to the format string, the log level is not a separate argument). The available log levels are:

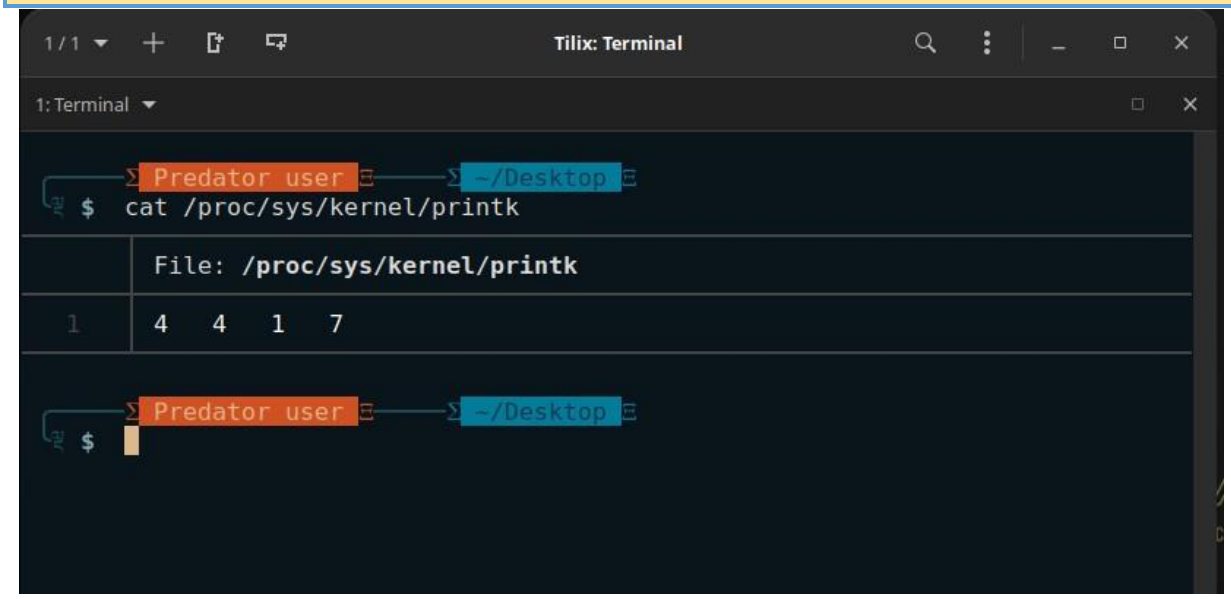
Name	String	Alias function
KERN_EMERG	"0"	pr_emerg()
KERN_ALERT	"1"	pr_alert()
KERN_CRIT	"2"	pr_crit()
		pr_err()

KERN_ERR	"3"	
KERN_WARNING	"4"	**pr_warn()**
KERN_NOTICE	"5"	**pr_notice()**
KERN_INFO	"6"	**pr_info()**
KERN_DEBUG	"7"	**pr_debug()** and **pr_devel()** if DEBUG is defined
KERN_DEFAULT	""	
KERN_CONT	"c"	**pr_cont()**

The log level specifies the importance of a message. The kernel decides whether to show the message immediately (printing it to the current console) depending on its log level and the current *console_loglevel* (a kernel variable). If the message priority is higher (lower log level value) than the *console_loglevel* the message will be printed to the console.

If the log level is omitted, the message is printed with KERN_DEFAULT level. You can check the current *console_loglevel* with:

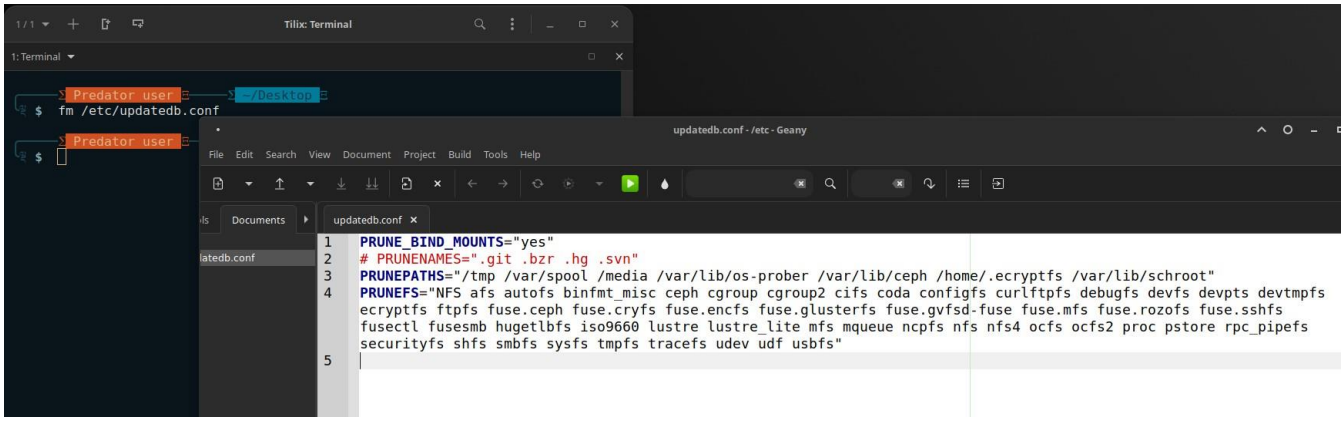
```
$ cat /proc/sys/kernel/printk
```



locate and updatedb

The **locate** command can find the location of a file when you only know part of the name. It sends a result almost instantaneously, since it consults a database that stores the location of all the files on the system; this database is updated daily by the

updatedb command. There are multiple implementations of the **locate** command and picked mlocate for its standard system. If you want to consider an alternative, you can try plocate which provides the same command line options and can be considered a drop-in replacement. **locate** is smart enough to only return files which are accessible to the user running the command even though it uses a database that knows about all files on the system (since its **updatedb** implementation runs with root rights). For extra safety, the administrator can use **PRUNEDPATHS** in **/etc/updatedb.conf** to exclude some directories from being indexed.



syslog System Events

Principle and Mechanism

The **rsyslogd** daemon is responsible for collecting service messages coming from applications and the kernel, then dispatching them into log files (usually stored in the **/var/log/** directory). It obeys the /etc/rsyslog.conf configuration file.

```
 # /etc/rsyslog.conf configuration file
for rsyslog #
# For more information install rsyslog-doc and see
# /usr/share/doc/rsyslog-doc/html/configuration/index.html
#
# Default logging rules can be found in /etc/rsyslog.d/50-default.conf

##################
#### MODULES ####
##################

module(load="imuxsock") # provides support for local system logging
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")

# provides kernel logging support and enable non-kernel klog messages
module(load="imklog" permitnonkernelfacility="on")

###############################
#### GLOBAL DIRECTIVES ####
###############################


#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# Filter duplicated messages
$RepeatedMsgReduction on

#
# Set the default permissions for all log files.
```

```
#
$FileOwner syslog
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog
```

```
#
# Where to place spool and state files
#
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
```

$IncludeConfig /etc/rsyslog.d/*.conf

Each log message is associated with an application subsystem (called "facility" in the documentation):

- **auth** and **authpriv**: for authentication;
- **cron**: comes from task scheduling services, **cron** and **atd**;
- **daemon**: affects a daemon without any special classification (DNS, NTP, etc.); ○ **ftp**: concerns the FTP server; ○ **kern**: message coming from the kernel; ○ **lpr**: comes from the printing subsystem; ○ **mail**: comes from the e-mail subsystem;
- **news**: Usenet subsystem message (especially from an NNTP — Network News
  Transfer Protocol — server that manages newsgroups); ○ **syslog**: messages from the **syslogd** server, itself;
- **user**: user messages (generic);
- **uucp**: messages from the UUCP server (Unix to Unix Copy Program, an old protocol notably used to distribute e-mail messages);
- **local0** to **local7**: reserved for local use.

Each message is also associated with a priority level. Here is the list in decreasing order:

- **emerg**: "Help!" There is an emergency, the system is probably unusable.
- **alert**: hurry up, any delay can be dangerous, action must be taken immediately; ○ **crit**: conditions are critical; ○ **err**: error;
- **warn**: warning (potential error);
- **notice**: conditions are normal, but the message is important; ○ **info**: informative message; ○ **debug**: debugging message.

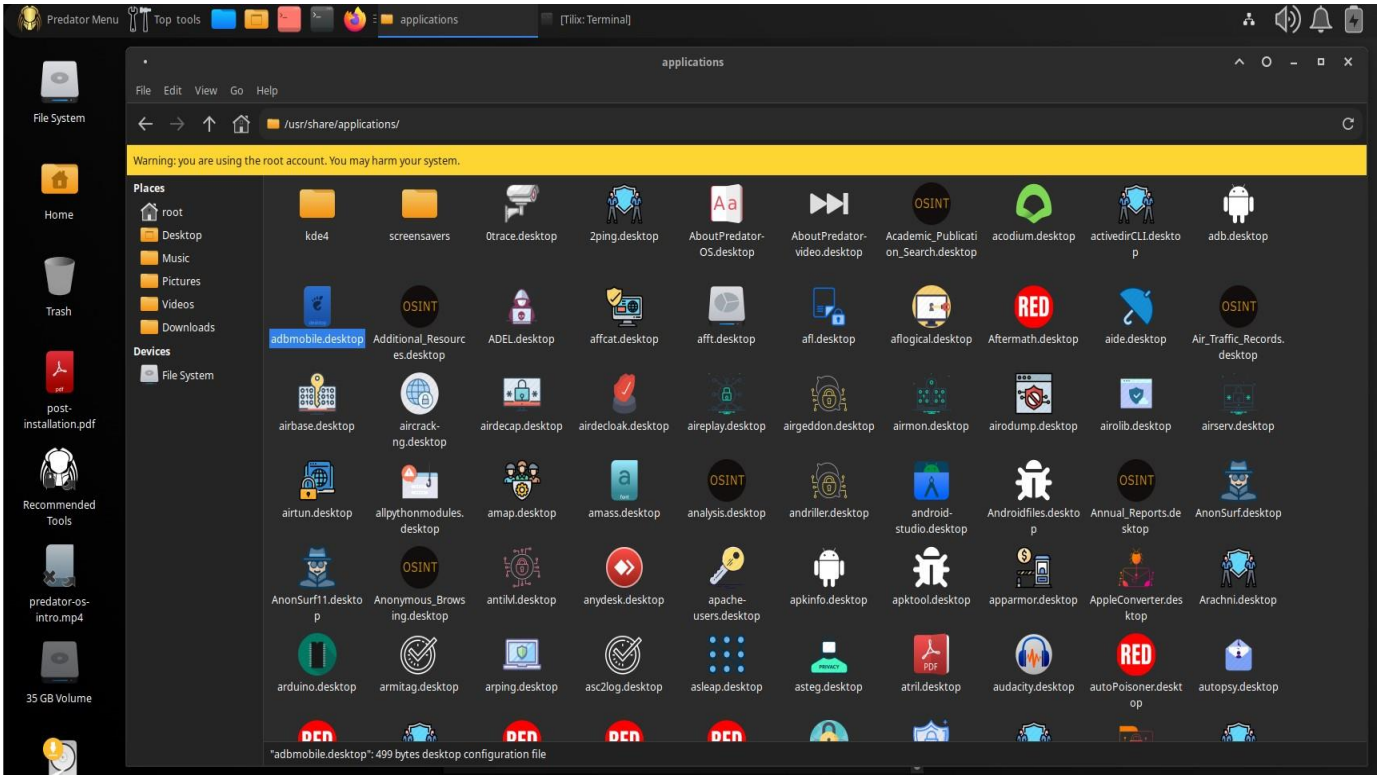
Plasma

Plasma is a simple and lightweight graphical desktop, which is a perfect match for computers with limited resources. It can be installed with **apt install Plasma4** (task-Plasma-desktop). Like GNOME, Plasma is based on the GTK+ toolkit, and several components are common across both desktops.

Unlike GNOME and Plasma, Plasma does not aim to become a vast project. Beyond the basic components of a modern desktop (file manager, window manager, session manager, a panel for application launchers and so on), it only provides a few specific applications: a terminal, a calendar (orage), an image viewer, a CD/DVD burning tool, a media player (parole), sound volume control and a text editor (mousepad).

Menu and tools overview

Modern desktop environments and many window managers provide menus listing the available applications for the user. In order to keep menus up-to-date in relation to the actual set of available applications, each package usually provides a **.desktop** file in **/usr/share/applications**:



Introduction to AppArmor

AppArmor is a Mandatory Access Control (MAC) system built on Linux's LSM (Linux Security Modules) interface. In practice, the kernel queries AppArmor before each system call to know whether the process is authorized to do the given operation. Through this mechanism, AppArmor confines programs to a limited set of resources. AppArmor applies a set of rules (known as "profile") on each program. The profile applied by the kernel depends on the installation path of the program being executed. Contrary to SELinux (discussed in Section 14.5, "Introduction to SELinux"), the rules applied do not depend on the user. All users face the same set of rules when they are executing the same program (but traditional user permissions still apply and might result in different behavior!).

AppArmor profiles are stored in **/etc/apparmor.d/** and they contain a list of access control rules on resources that each program can make use of. The profiles are compiled and loaded into the kernel by the **apparmor_parser** command. Each profile can be loaded either in enforcing or complaining mode. The former enforces the policy and reports violation attempts, while the latter does not enforce the policy but still logs the system calls that would have been denied.

AppArmor

AppArmor is a product of Canonical, Ltd., releasers of the Debian stable distribution.

It's supported by Debian and Debian stable, but has also been adopted as a standard by SUSE distributions. Debian stable and SUSE enable it on default installs, although the complement of protected services is not extensive.

AppArmor implements a form of MAC and is intended as a supplement to the traditional UNIX access control system. Although any configuration is possible, AppArmor is not designed to be a user-facing system. Its main goal is service securement; that is, limiting the damage that individual programs can do if they should be compromised or run amok.

Protected programs continue to be subject to all the limitations imposed by the standard model, but in addition, the kernel filters their activities through a designated and task-specific AppArmor profile. By default, AppArmor denies all requests, so the profile must explicitly name everything the process is allowed to do.

Programs without profiles, such as user shells, have no special restrictions and run as if AppArmor were not installed.

This service securement role is essentially the same configuration that's implemented by SELinux in Red Hat's targeted environment. However, AppArmor

is designed more specifically for service securement, so it sidesteps some of the more puzzling nuances of SELinux.

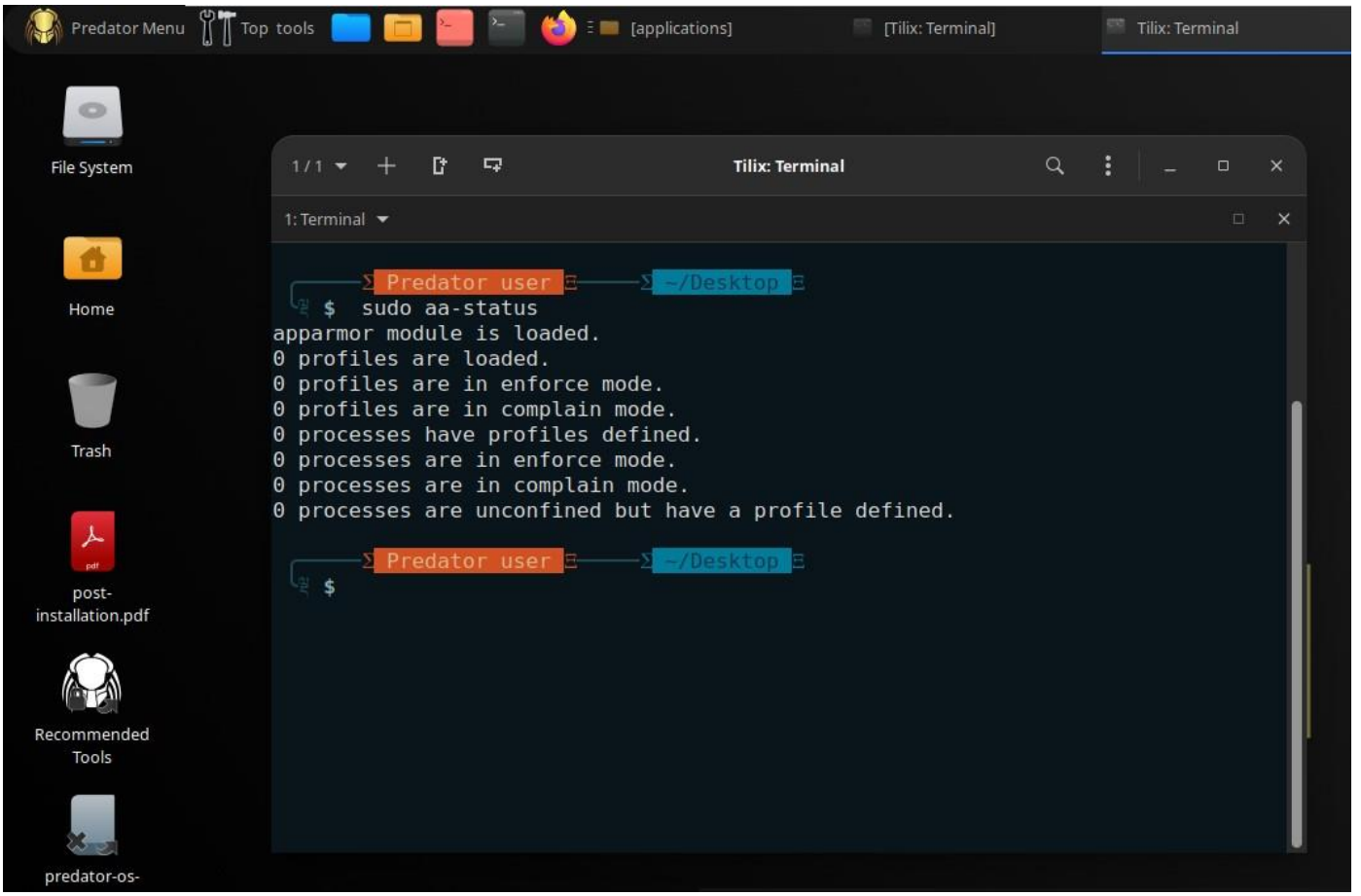
AppArmor profiles are stored in /etc/apparmor.d, and they are relatively readable even without detailed knowledge of the system.

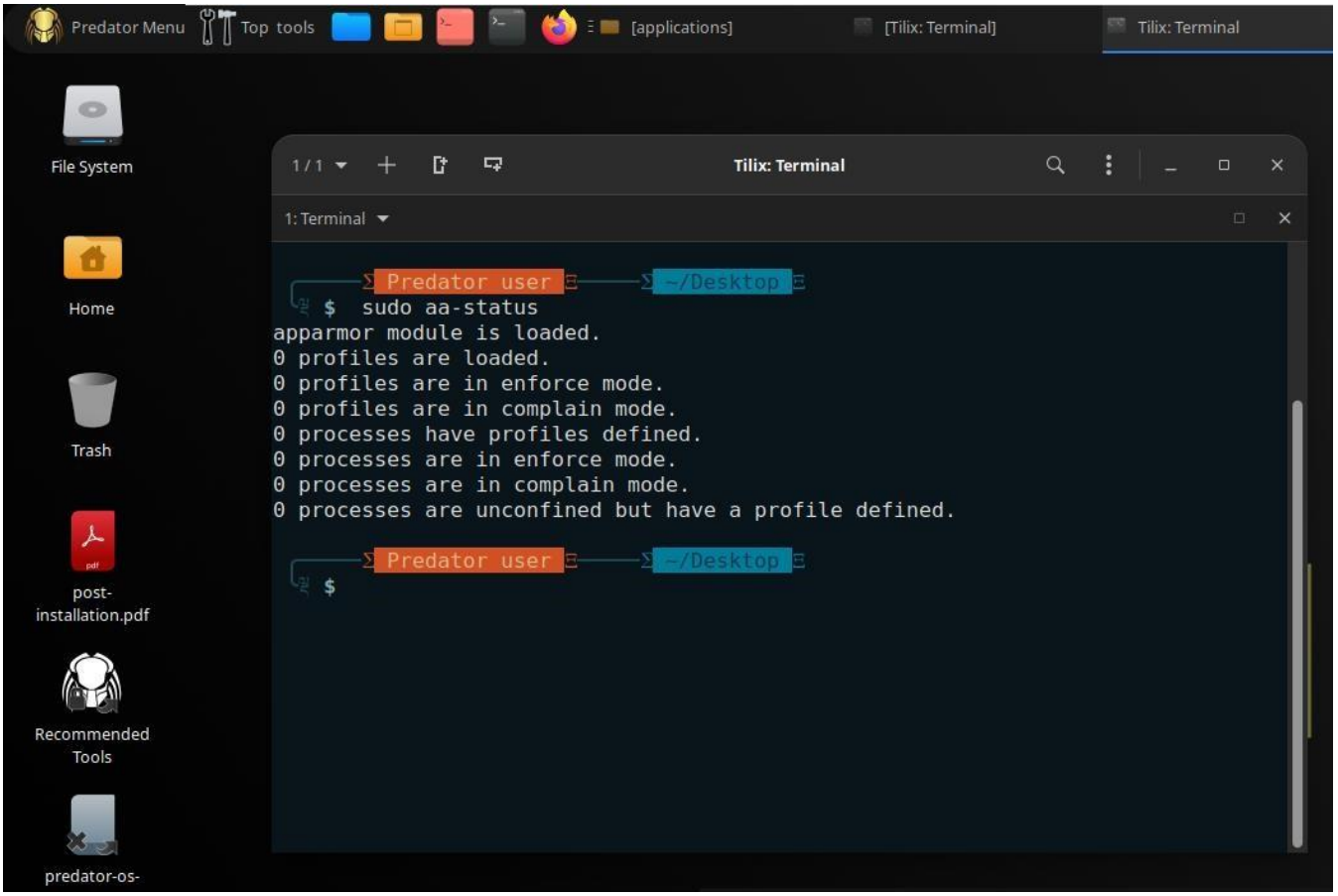
Enabling AppArmor and managing AppArmor profiles

AppArmor support is built into the standard kernels provided by Debian. Enabling AppArmor is thus just a matter of installing some packages by executing **apt install apparmor apparmor-profiles apparmor-utils** with root privileges.

AppArmor is functional after the installation, and **aa-status** will confirm it quickly:

```
# aa-status
```


Predator Menu
Top tools
[applications]
[Tilix: Terminal]
Tilix: Terminal
File System
Home
Trash
post-installation.pdf
Recommended Tools
predator-os-
1 / 1
Tilix: Terminal
1: Terminal
Predator user
~/Desktop
$ sudo aa-status
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
Predator user
~/Desktop
$


Predator Menu
Top tools
[applications]
[Tilix: Terminal]
Tilix: Terminal
File System
Home
Trash
post-installation.pdf
Recommended Tools
predator-os-
1 / 1
Tilix: Terminal
1: Terminal
Predator user
~/Desktop
$ sudo aa-status
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
Predator user
~/Desktop
$

SELinux (*Security Enhanced Linux*) is a *Mandatory Access Control* system built on Linux's LSM (*Linux Security Modules*) interface. In practice, the kernel queries SELinux before each system call to know whether the process is authorized to do the given operation.

SELinux uses a set of rules — collectively known as a *policy* — to authorize or forbid operations. Those rules are difficult to create. Fortunately, two standard policies (*targeted* and *strict*) are provided to avoid the bulk of the configuration work.

With SELinux, the management of rights is completely different from traditional Unix systems. The rights of a process depend on its *security context*. The context is defined by the *identity* of the user who started the process, the *role* and the *domain* that the user carried at that time. The rights really depend on the domain, but the transitions between domains are controlled by the roles. Finally, the possible transitions between roles depend on the identity.

Setting Up SELinux

SELinux support is built into the standard kernels provided by Debian. The core Unix tools support SELinux without any modifications. It is thus relatively easy to enable SELinux.

The **apt install selinux-basics selinux-policy-defaulti auditd** command will automatically install the packages required to configure an SELinux system.

The selinux-policy-default package contains a set of standard rules. By default, this policy only restricts access for a few widely exposed services. The user sessions are not restricted and it is thus unlikely that SELinux would block legitimate user operations.

Modern access control

Given the world's wide range of computing environments and the mixed success of efforts to advance the standard model, kernel maintainers have been reluctant to act as mediators in the larger debate over access control. In the Linux world, the situation came to a head in 2001, when the U.S. National Security Agency proposed to integrate its Security-Enhanced Linux (SELinux) system into the kernel as a standard facility.

For several reasons, the kernel maintainers resisted this merge. Instead of adopting SELinux or another, alternative system, they developed the Linux Security Modules API, a kernel-level interface that allows access control systems to integrate themselves as loadable kernel modules.

LSM-based systems have no effect unless users load them and turn them on. This fact lowers the barriers for inclusion in the standard kernel, and Linux now ships with SELinux and four other systems (AppArmor, Smack, TOMOYO, and Yama) ready to go.

Developments on the BSD side have roughly paralleled those of Linux, thanks largely to Robert Watson's work on TrustedBSD. This code has been included in FreeBSD since version 5. It also provides the application sandboxing technology used in Apple's macOS and iOS.

When multiple access control modules are active simultaneously, an operation must be approved by all of them to be permitted. Unfortunately, the LSM system requires explicit cooperation among active modules, and none of the current modules include this feature. For now, Linux systems are effectively limited to a choice of one LSM add-on module

SELinux: Security-Enhanced Linux

SELinux is one of the oldest Linux MAC implementations and is a product of the U.S. National Security Agency. Depending on one's perspective, that might be a source of either comfort or suspicion.7

SELinux takes a maximalist approach, and it implements pretty much every flavor of MAC and RBAC one might envision. Although it has gained footholds in a few distributions, it is notoriously difficult to administer and troubleshoot. This unattributed quote from a former version of the SELinux Wikipedia page vents the frustration felt by many sysadmins:

Intriguingly, although the stated raison d'être of SELinux is to facilitate the creation of individualized access control policies specifically attuned to organizational data custodianship practices and rules, the supportive software tools are so sparse and unfriendly that the vendors survive chiefly on "consulting," which typically takes the form of incremental modifications to boilerplate security policies.

Despite its administrative complexity, SELinux adoption has been slowly growing, particularly in environments such as government, finance, and health care that enforce strong and specific security requirements. It is also a standard part of the Android platform.

Our general opinion regarding SELinux is that it is capable of delivering more harm than benefit. Unfortunately, that harm can manifest not only as wasted time and as aggravation for system administrators, but ironically, as security lapses. Complex models are hard to reason about, and SELinux is not really a level playing field; hackers that focus on it understand the system far more thoroughly than the average sysadmin.

In particular, SELinux policy development is a complicated endeavor. To protect a new daemon, for example, a policy must carefully enumerate all the files, directories, and other objects to which the process needs access. For complicated software like sendmail or httpd, this task can be quite complex. At least one company offers a three-day class on policy development.

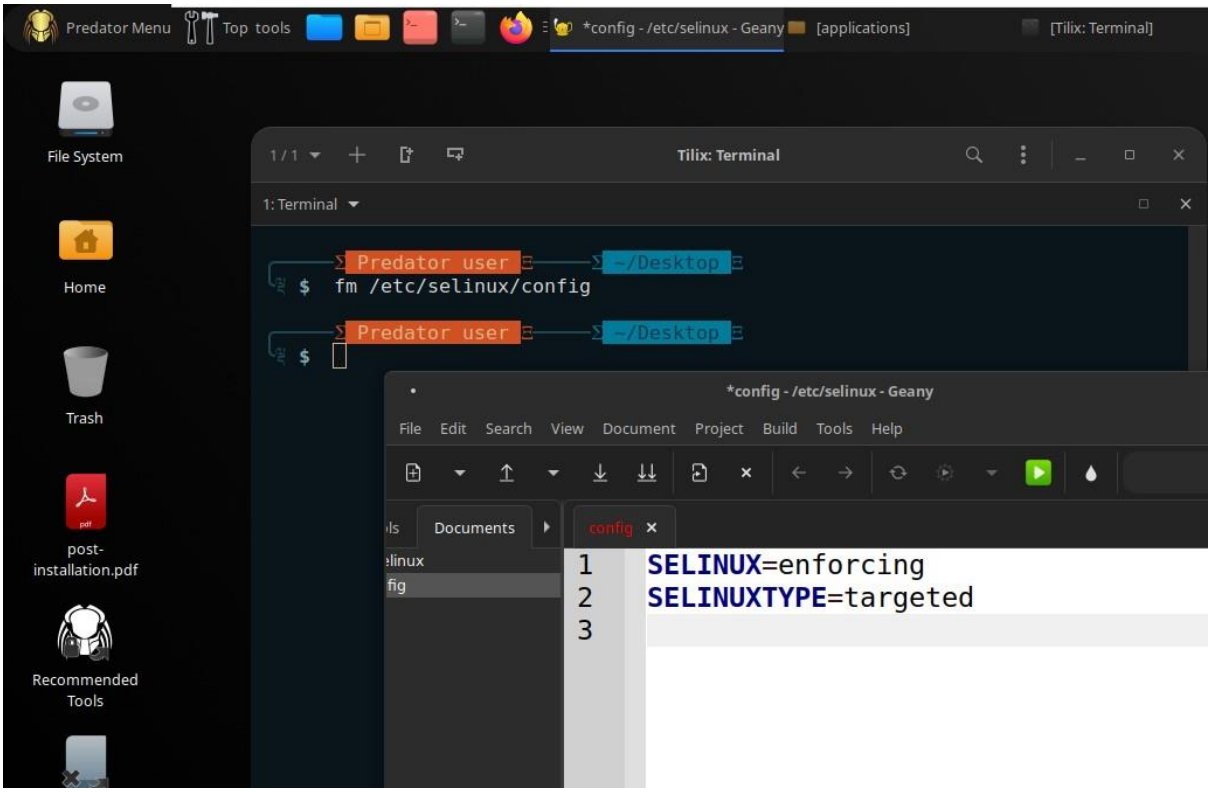
Fortunately, many general policies are available on-line, and most SELinux-enabled distributions come with reasonable defaults. These can easily be installed and configured for your particular environment. A full-blown policy editor that aims to ease policy application can be found at seedit.sourceforge.net.

SELinux is well supported by both Red Hat (and hence, CentOS) and Fedora. Red Hat enables it by default.

Debian and SUSE Linux also have some available support for SELinux, but you must install additional packages, and the system is less aggressive in its default configuration.

Debian stable inherits some SELinux support from Debian, but over the last few releases, Debian stable's focus has been on AppArmor (see page 87). Some vestigial SELinux-related packages are still available, but they are generally not up to date. **/etc/selinux/config** is the top-level control for SELinux. The interesting lines are

```
SELINUX=enforcing
SELINUXTYPE=targeted
```



The first line has three possible values: enforcing, permissive, or disabled. The enforcing setting ensures that the loaded policy is applied and prohibits violations. permissive allows violations to occur but logs them through syslog, which is valuable for debugging and policy development. disabled turns off SELinux entirely.

SELINUXTYPE refers to the name of the policy database to be applied. This is essentially the name of a subdirectory within /etc/selinux. Only one policy can be active at a time, and the available policy sets vary by system.

Managing an SELinux System

The SELinux policy is a modular set of rules, and its installation detects and enables automatically all the relevant modules based on the already installed services. The system is thus immediately operational. However, when a service is installed after the SELinux policy, you must be able to manually enable the corresponding module. That is the purpose of the **semodule** command. Furthermore, you must be able to define the roles that each user can endorse, and this can be done with the **semanage** command.

Those two commands can thus be used to modify the current SELinux configuration, which is stored in **/etc/selinux/default/**. Unlike other configuration files that you can find in **/etc/**, all those files must not be changed by hand. You should use the programs designed for this purpose.

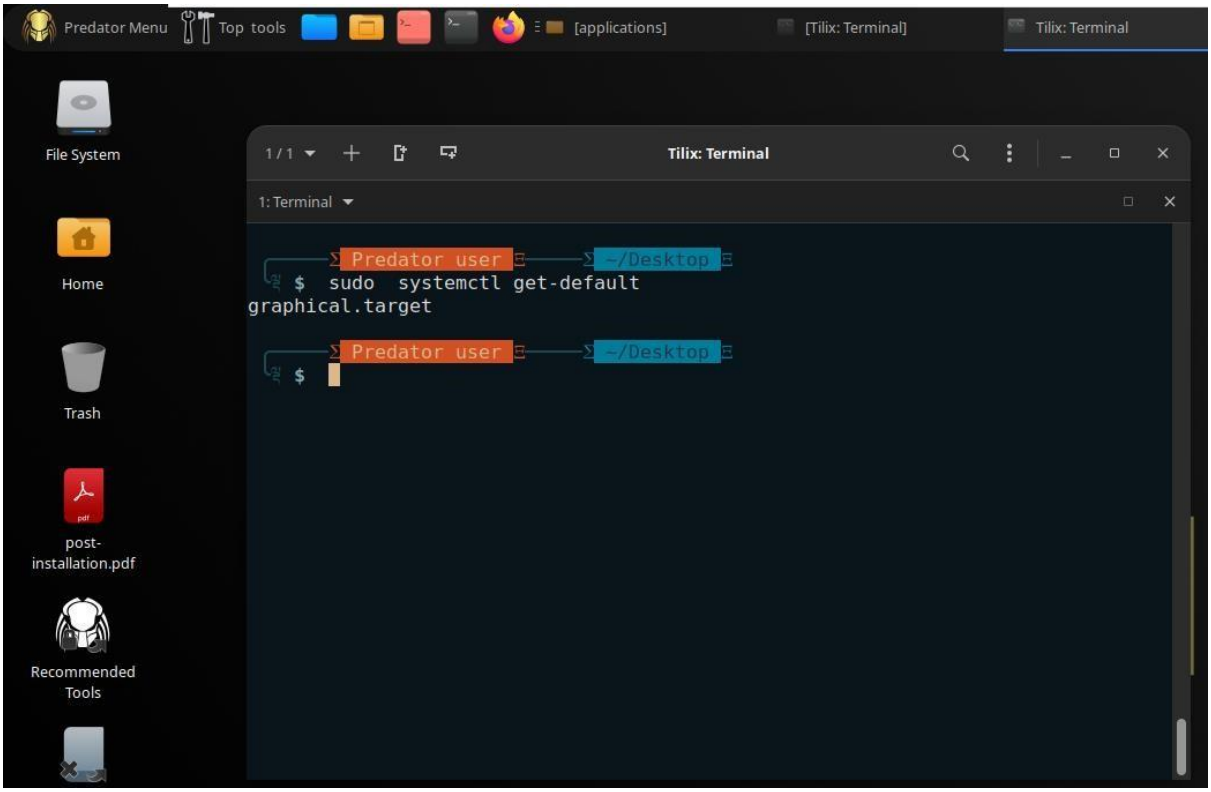
systemd

systemd is a suite of basic building blocks for a Linux system. It provides a system and service manager that runs as PID 1 and starts the rest of the system. systemd provides aggressive parallelization capabilities, uses socket and DBus activation for starting services, offers on-demand starting of daemons, keeps track of processes using Linux control groups, maintains mount and automount points, and implements an elaborate transactional dependency-based service control logic. systemd supports SysV and LSB init scripts and works as a replacement for sysvinit. Other parts include a logging daemon, utilities to control basic system configuration like the hostname, date, locale, maintain a list of logged-in users and running containers and virtual machines, system accounts, runtime directories and settings, and daemons to manage simple network configuration, network time synchronization, log forwarding, and name resolution. systemd in detail The configuration and control of system services is an area in which Linux distributions have traditionally differed the most from one another. systemd aims to standardize this aspect of system administration, and to do so, it reaches further into the normal operations of the system than any previous alternative.

To see the target the system boots into by default, run the **get-default** subcommand:

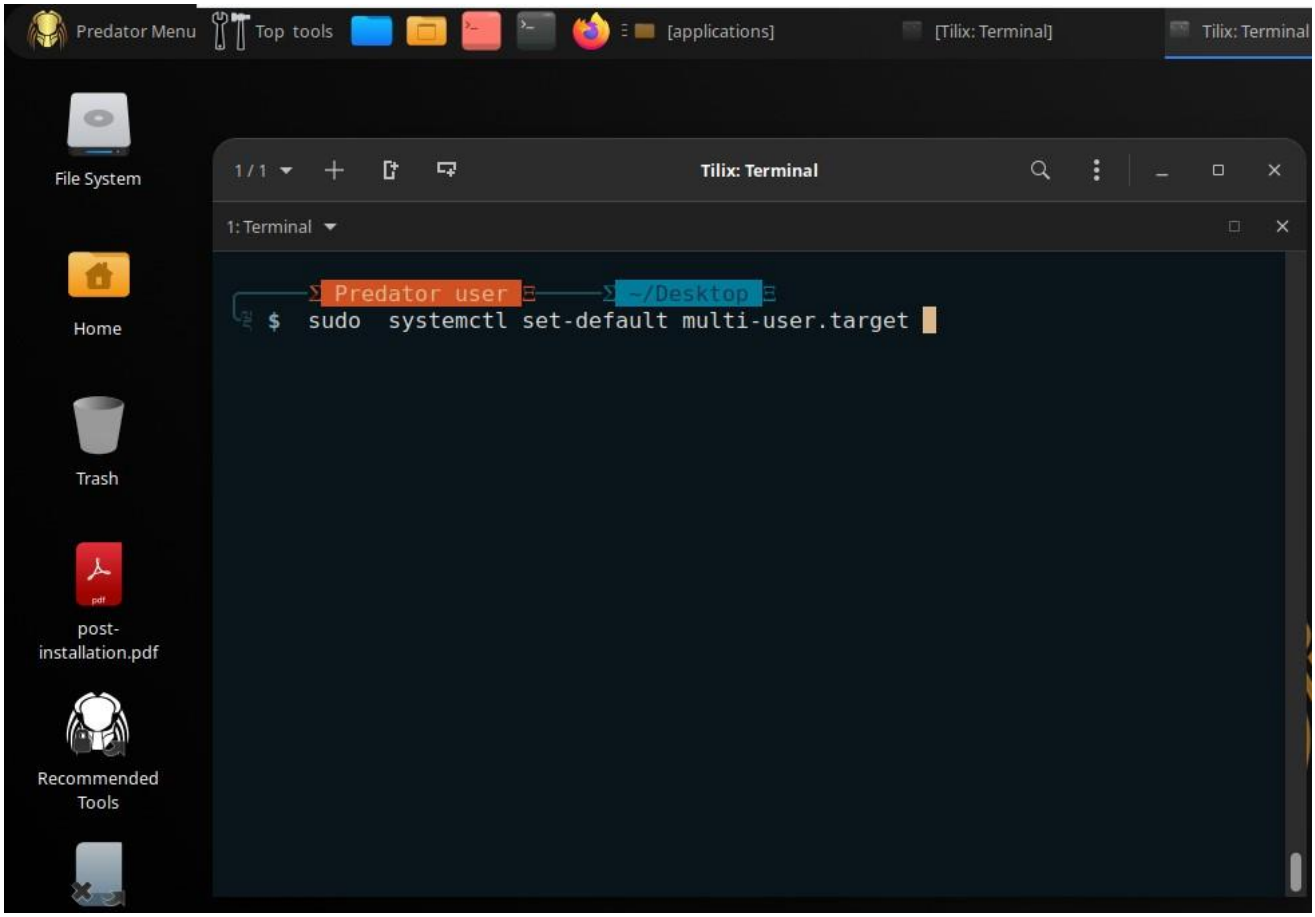
```
$ systemctl get-default
```

graphical.target



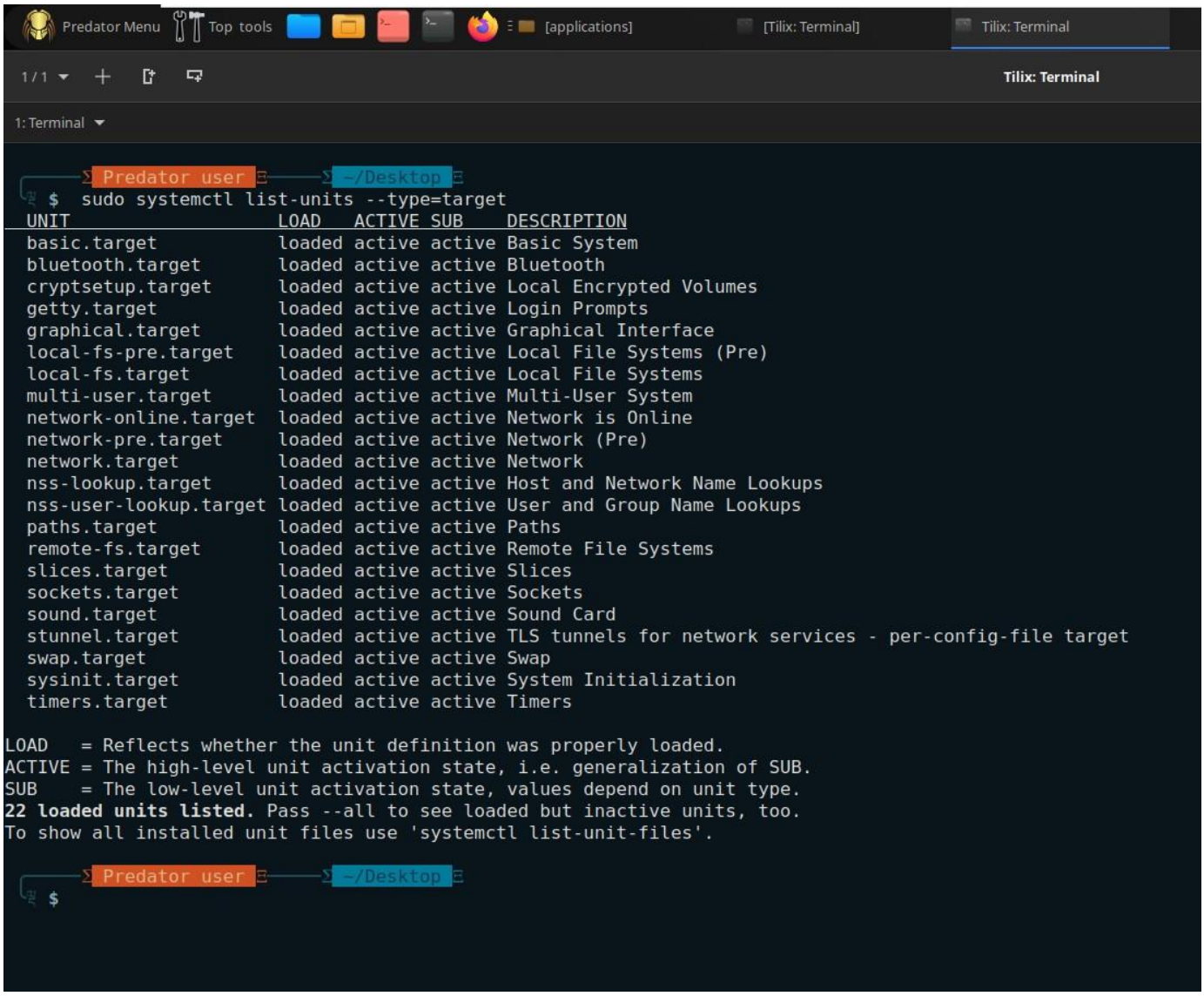
Most Linux distributions boot to **graphical.target** by default, which isn't appropriate for servers that don't need a GUI. But that's easily changed:

```
$ sudo systemctl set-default multi-user.target
```



To see all the system's available targets, run **systemctl list-units**:

```
$ systemctl list-units --type=target
```



systemctl

Activates a service immediately:

```
systemctl start your_service.service
```

Deactivates a service immediately:

```
systemctl stop your_service.service
```

Restarts a service:

```
systemctl restart your_service.service
```

Shows status of a service including whether it is running or not:

```
systemctl status your_service.service
```

Enables a service to be started on bootup:

```
systemctl enable your_service.service
```

Disables a service to not start during bootup:

```
systemctl disable your_service.service
```

Microcode

Processor manufacturers release stability and security updates to the processor **microcode**. These updates provide bug fixes that can be critical to the stability of your system. Without them, you may experience spurious crashes or unexpected system halts that can be difficult to track down. All users with an AMD or Intel CPU should install the microcode updates to ensure system stability. To acquire updated microcode, depending on the processor, pre-installed the following packages on Predator-OS:

```
amd-ucode for AMD processors, intel-ucode for Intel processors.
```

Power management with systemd

ACPI events

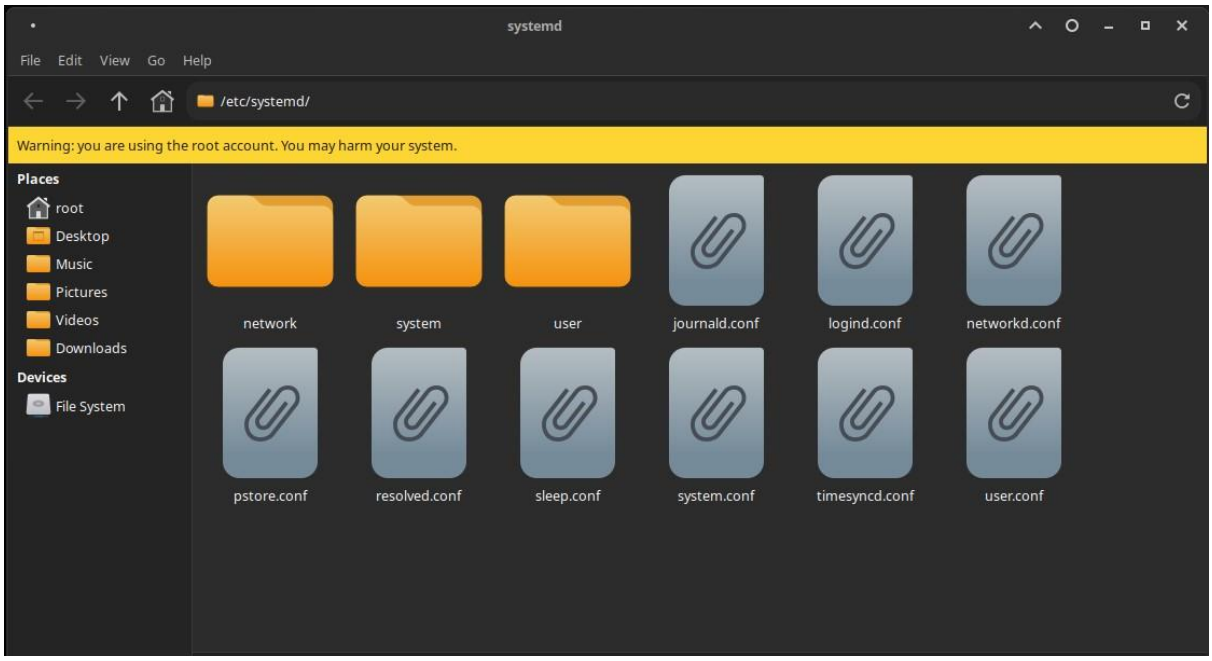
systemd handles some power-related ACPI events, whose actions can be configured in /etc/systemd/logind.conf or /etc/systemd/logind.conf.d/*.conf .

```
source of : /etc/systemd/logind.conf
```

```
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by #  the Free Software Foundation; either version 2.1 of the License, or #  (at your option) any later version.
#
# Entries in this file show the compile time defaults.
```

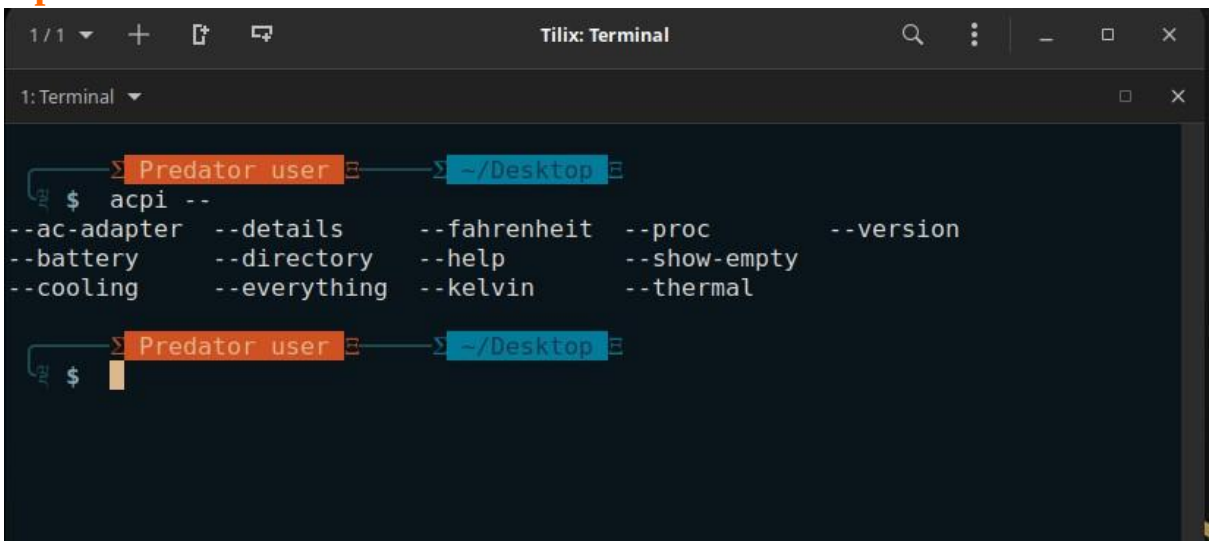
```
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file.
#
# See logind.conf(5) for details.

[Login]
#NAutoVTs=6
#ReserveVT=6
#KillUserProcesses=no
#KillOnlyUsers=
#KillExcludeUsers=root
#InhibitDelayMaxSec=5
#UserStopDelaySec=10
#HandlePowerKey=poweroff
#HandleSuspendKey=suspend
#HandleHibernateKey=hibernate
#HandleLidSwitch=suspend
#HandleLidSwitchExternalPower=suspend
#HandleLidSwitchDocked=ignore
#HandleRebootKey=reboot
#PowerKeyIgnoreInhibited=no
#SuspendKeyIgnoreInhibited=no
#HibernateKeyIgnoreInhibited=no
#LidSwitchIgnoreInhibited=yes
#RebootKeyIgnoreInhibited=no
#HoldoffTimeoutSec=30s
#IdleAction=ignore
#IdleActionSec=30min
#RuntimeDirectorySize=10%
#RuntimeDirectoryInodes=400k
#RemoveIPC=yes
#InhibitorsMax=8192 #SessionsMax=8192
```

```
/etc/systemd/
```

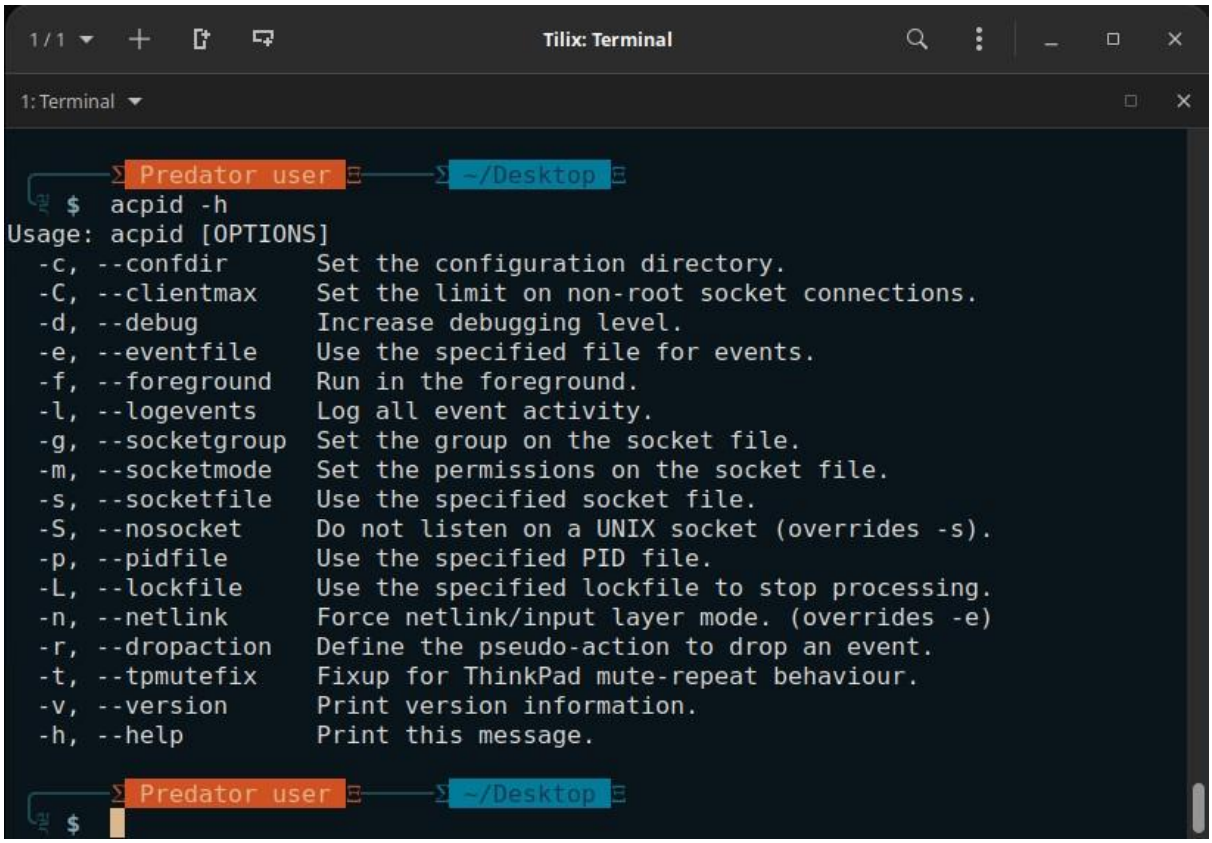


On systems with no dedicated power manager, this may replace the **acpid** daemon which is usually used to react to these ACPI events.

acpi



acpid command



The specified action for each event can be one of ignore, poweroff, reboot, halt, suspend, hibernate, hybrid-sleep, suspend-thenhibernate, lock or kexec. In case of hibernation and suspension, they must be properly set up. If an event is not configured, systemd will use a default action.

Event handler	Description	Default action
HandlePowerKey	Triggered when the power key/button is pressed.	poweroff
HandleSuspendKey	Triggered when the suspend key/button is pressed.	suspend
HandleHibernateKey	Triggered when the hibernate key/button is pressed.	hibernate
HandleLidSwitch	Triggered when the lid is closed, except in the cases below.	suspend

HandleLidSwitchDocked	Triggered when the lid is closed if the system is inserted in a docking station, or more than one display is connected.	ignore
HandleLidSwitchExternalPower	Triggered when the lid is closed if the system is connected to external power.	action set for HandleLidSwitch

To apply any changes, signal :

```
# systemctl kill -s HUP systemd-logind
```

By default, these features are Disabled in the Predator-OS:

```
sudo systemctl disable  sleep.target suspend.target hibernate.target hybrid-sleep.target sudo systemctl stop  sleep.target suspend.target hibernate.target hybrid-sleep.target
```

Bluetooth

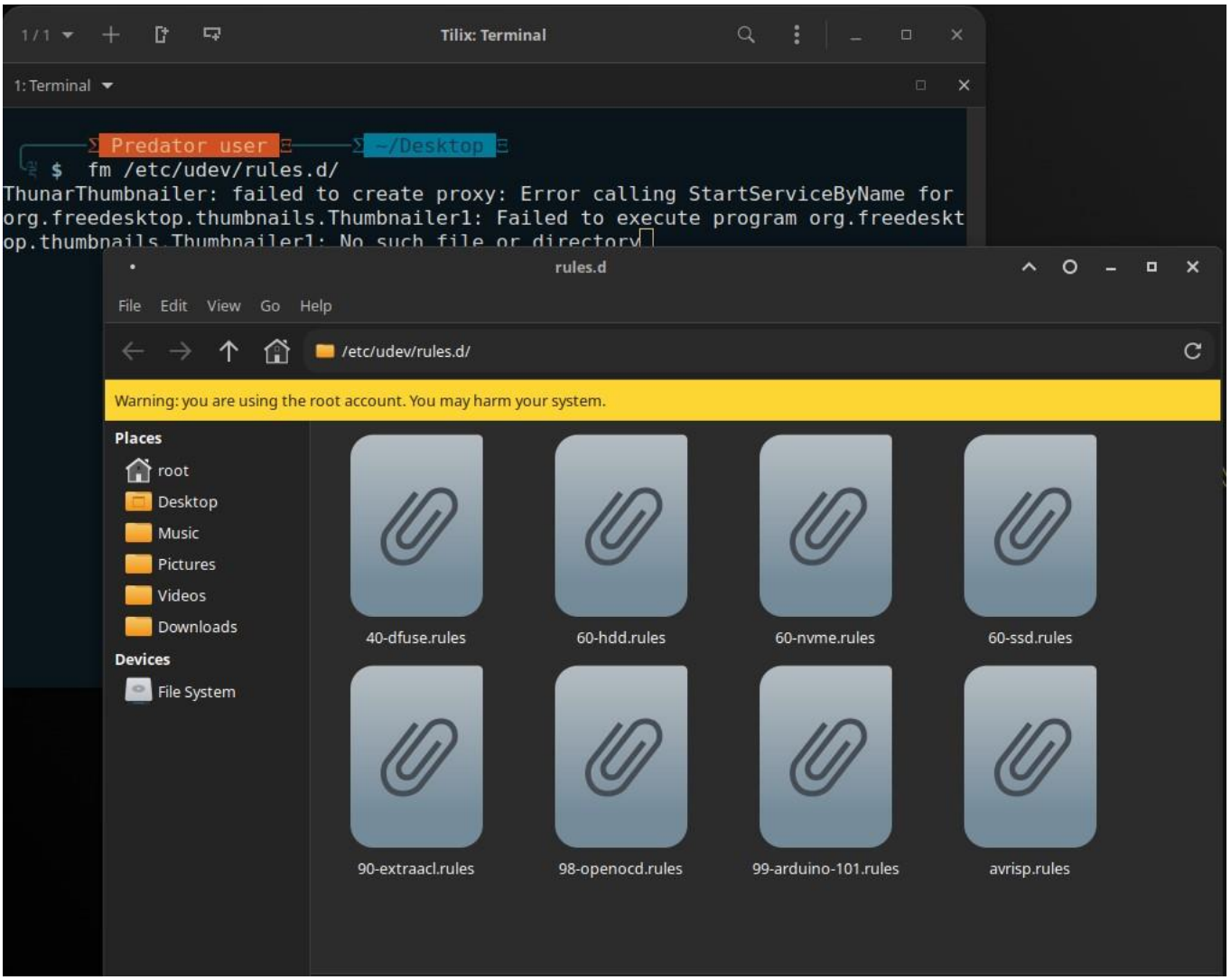
To disable bluetooth completely, **blacklist** the `btusb` and `bluetooth` modules.

To turn off bluetooth only temporarily, use *rfkill*:

```
# rfkill block bluetooth
```

Or with udev rule:

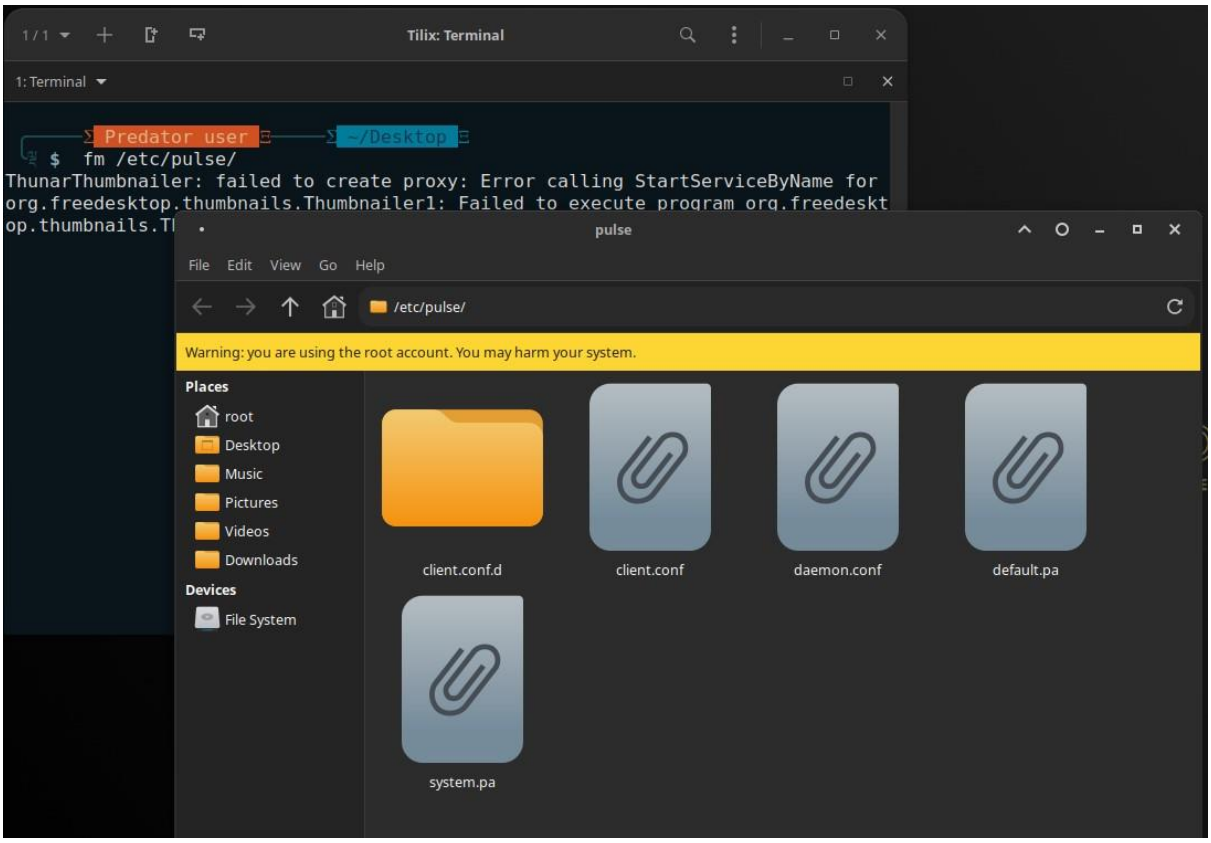
```
/etc/udev/rules.d/50-bluetooth.rules
# disable bluetooth
SUBSYSTEM==“rfkill”, ATTR{type}==“bluetooth”, ATTR{state}=“0”
```



PulseAudio

By default, PulseAudio suspends any audio sources that have become idle for too long. When using an external USB microphone, recordings may start with a pop sound. As a workaround, comment `/etc/pulse/default.pa` out the following line in

```
### Automatically suspend sinks/sources that become idle for too long load-module module-suspend-on-idle
```


1 / 1
Tilix: Terminal
1: Terminal
Predator user
~/Desktop
$ fm /etc/pulse/
ThunarThumbnailer: failed to create proxy: Error calling StartServiceByName for
org.freedesktop.thumbnails.Thumbnailer1: Failed to execute program org.freedeskt
op.thumbnails.T
pulse
File Edit View Go Help
/etc/pulse/
Warning: you are using the root account. You may harm your system.
Places
root
Desktop
Music
Pictures
Videos
Downloads
Devices
File System
client.conf.d
client.conf
daemon.conf
default.pa
system.pa

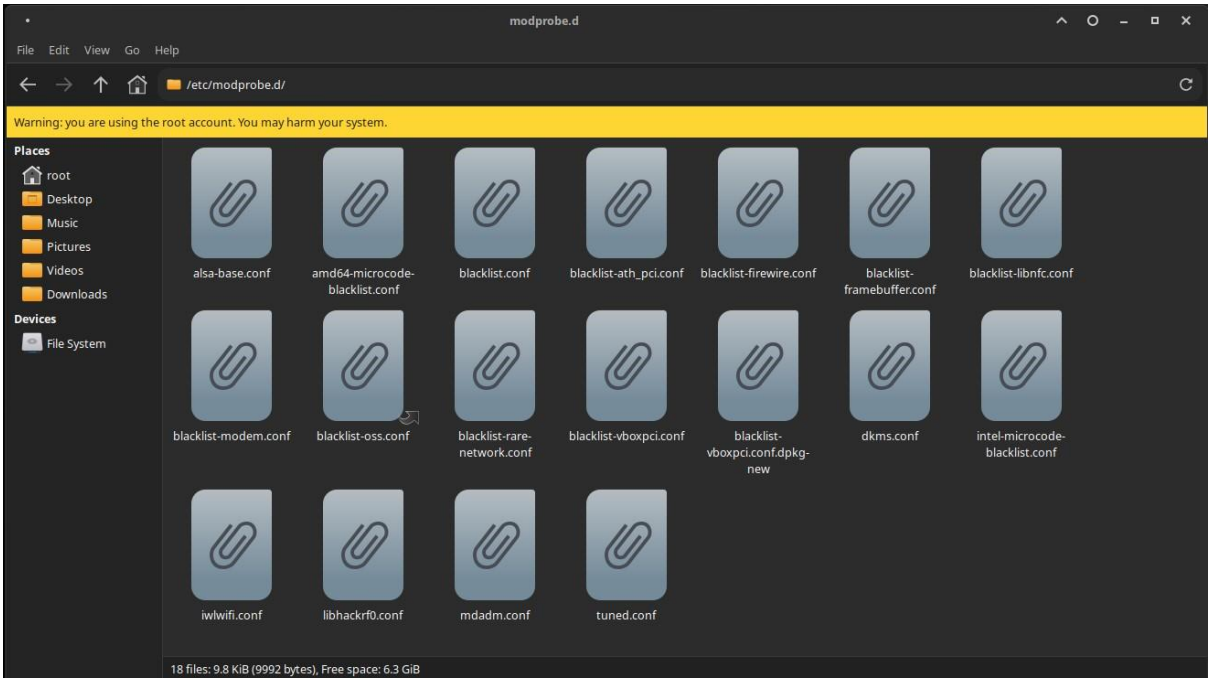
Blacklist Unneeded Modules

Modules can hog memory and may slow down your system. You can list all the modules currently required by your system by issuing `lsmod` command as regular or root user. Blacklist modules that you don't need.

The Linux kernel is modular, which makes it more flexible than monolithic kernels. New functionality can be easily added to a run kernel, by loading the related module. While that is great, it can also be misused. You can think of loading malicious modules (e.g. rootkits), or unauthorized access to the server and copy data via a USB port. In our previous article about kernel modules, we looked at how to prevent loading any module. In this case, we specifically disallow the ones we don't want.

Blacklisting modules

Blacklisting modules is one way to disallow them. This defines which modules should no longer be loaded. However, it will only limit the loading of modules during the boot process. You can still load a module manually after booting. Blacklisting a module is simple. Create a file in the /etc/modprobe.d directory and give it a proper name (e.g. blacklist-module.conf).



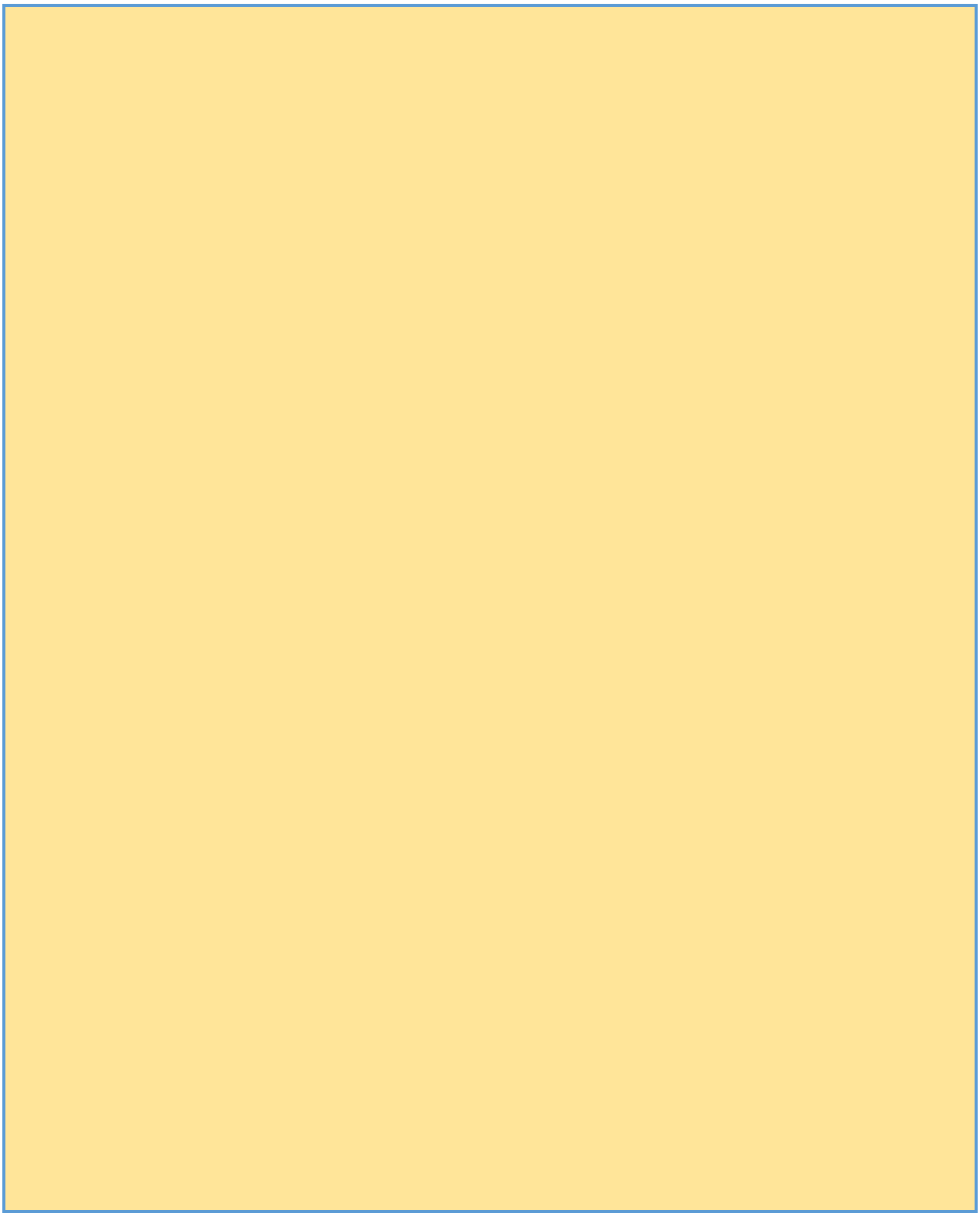
For me the /etc/modprobe.d/blacklist.conf goes like this:

```
blacklist iTCO_wdt
blacklist pcspkr
blacklist joydev
blacklist mousedev
```

```
blacklist mac_hid
blacklist uvcvideo
```

source of /etc/modprobe.d/blacklist.conf

```
# This file lists those modules which we don't want to be loaded by #
alias expansion, usually so some other driver will be loaded for the #
device instead.

# evbug is a debug tool that should be loaded explicitly blacklist
evbug

# these drivers are very simple, the HID drivers are usually preferred
#blacklist usbmouse
#blacklist usbkbd

# replaced by e100
blacklist eepro100

# replaced by tulip
blacklist de4x5

# causes no end of confusion by creating unexpected network interfaces blacklist
eth1394

# snd_intel8x0m can interfere with snd_intel8x0, does not seem to support much
# hardware on its own (Debian stable bug #2011, #6810) blacklist
snd_intel8x0m

# Conflicts with dvb driver (which is better for handling this device) blacklist
snd_aw2

# replaced by p54pci
blacklist prism54

# replaced by b43 and ssb.
blacklist bcm43xx

# most apps now use garmin usb driver directly (Debian stable: #114565) blacklist
garmin_gps
```



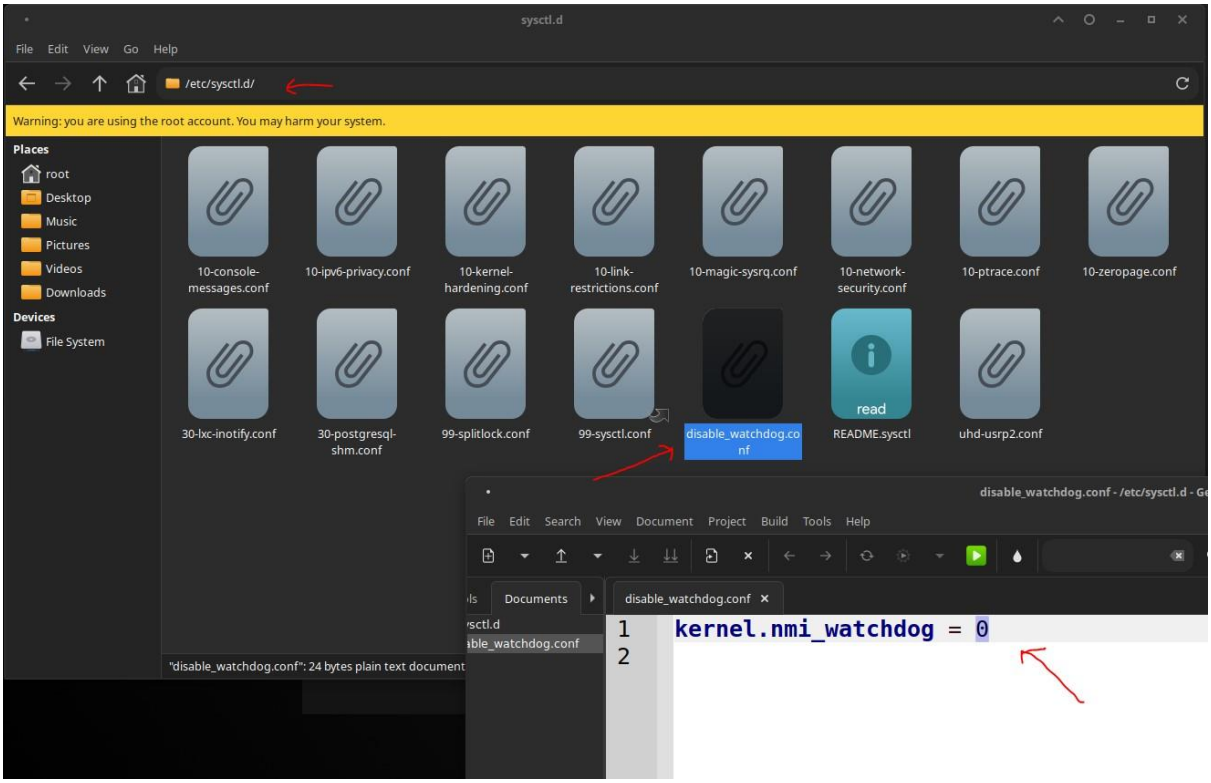
```
# replaced by asus-laptop (Debian stable: #184721) blacklist
asus_acpi

# low-quality, just noise when being used for sound playback, causes
# hangs at desktop session start (Debian stable: #246969) blacklist
snd_pcsp

# ugly and loud noise, getting on everyone's nerves; this should be done by a
# nice pulseaudio bing (Debian stable: #77010) blacklist
pcspkr

# EDAC driver for amd76x clashes with the agp driver preventing the aperture
# from being initialised (Debian stable: #297750). Blacklist so that the driver
# continues to build and is installable for the few cases where its # really
needed.
blacklist amd76x_edac

blacklist iTCO_wdt blacklist
joydev
```

Disabling NMI watchdog

The **NMI** watchdog is a debugging feature to catch hardware hangs that cause a kernel panic. On some systems it can generate a lot of interrupts, causing a noticeable increase in power usage:

```
/etc/sysctl.d/disable_watchdog.conf kernel.nmi_watchdog
= 0
```

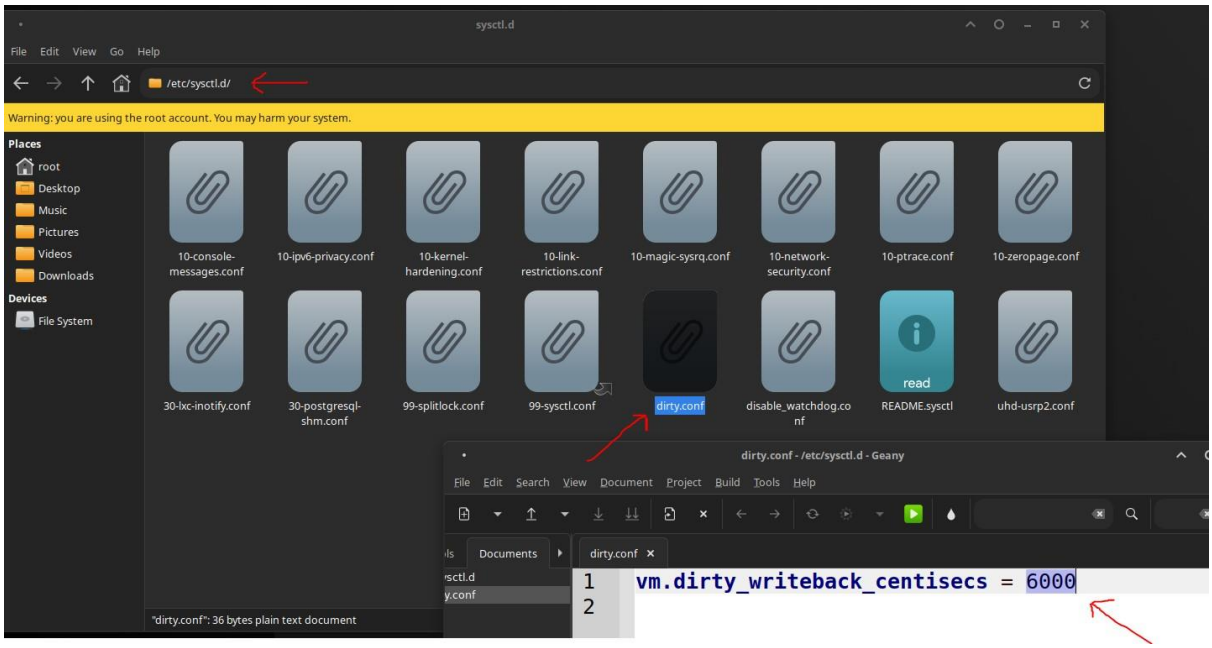
or add `nmi_watchdog=0` to the **kernel line** to disable it completely from early boot.



Writeback Time

Increasing the virtual memory dirty writeback time helps to aggregate disk I/O together, thus reducing spanned disk writes, and increasing power saving. To set the value to 60 seconds (default is 5 seconds):

```
/etc/sysctl.d/dirty.conf
vm.dirty_writeback_centisecs = 6000
```



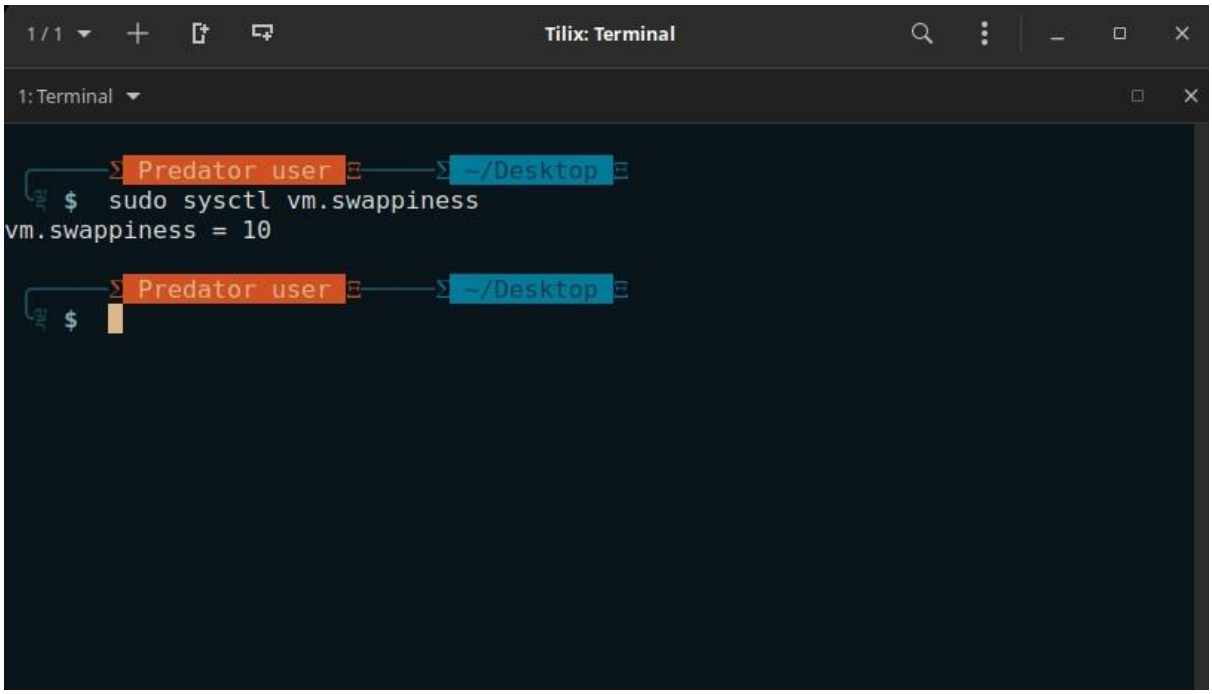
To do the same for journal commits on supported filesystems (e.g. ext4, btrfs...), use `commit=60` as a option in **fstab**.

Swappiness

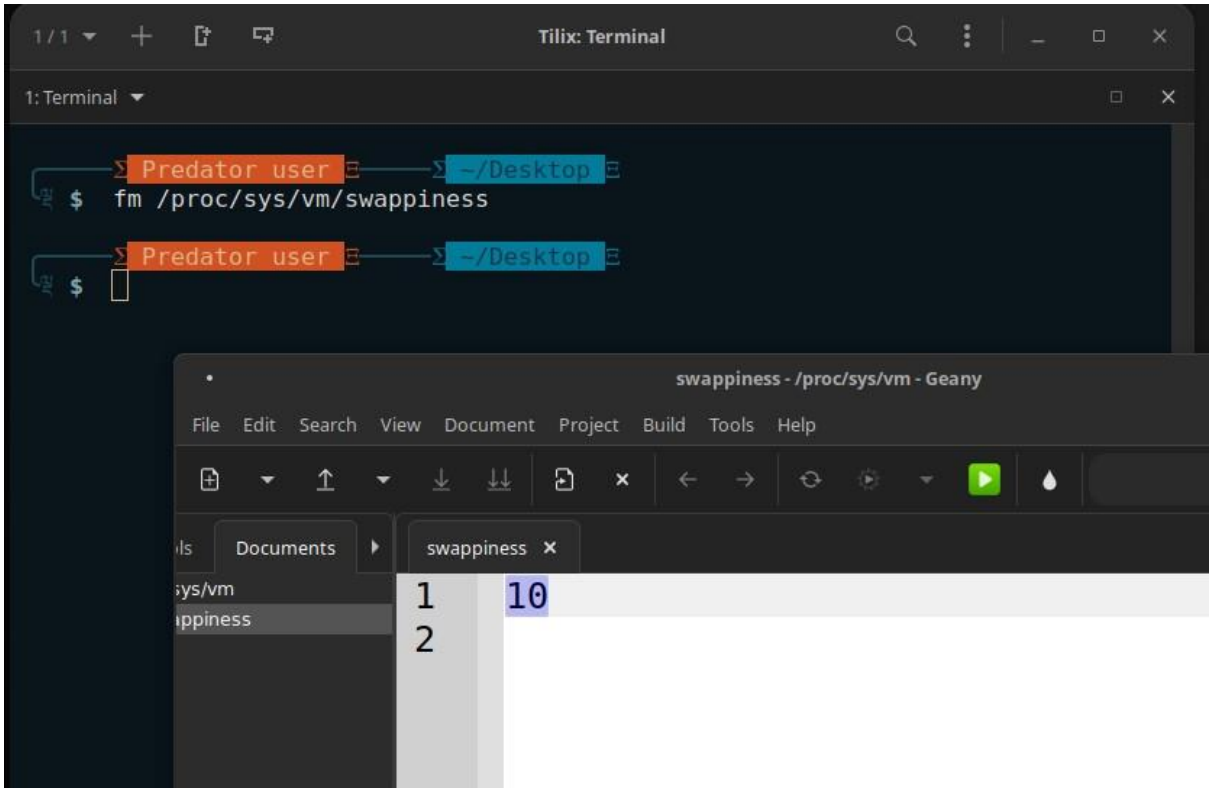
The *swappiness* **sysctl** parameter represents the kernel's preference (or avoidance) of swap space. Swappiness can have a value between 0 and 200 (max 100 if Linux < 5.8), the default value is 60. A low value causes the kernel to avoid swapping, a high value causes the kernel to try to use swap space, and a value of 100 means IO cost is assumed to be equal. Using a low value on sufficient memory is known to improve responsiveness on many systems.

To check the current swappiness value:

```
$ sysctl vm.swappiness
```



Alternatively, the files /sys/fs/cgroup/memory/memory.swappiness (cgroup v1specific) or /proc/sys/vm/swappiness can be read in order to obtain the raw integer value.



To temporarily set the swappiness value:

```
# sysctl -w vm.swappiness=10
```

Using zswap or zram

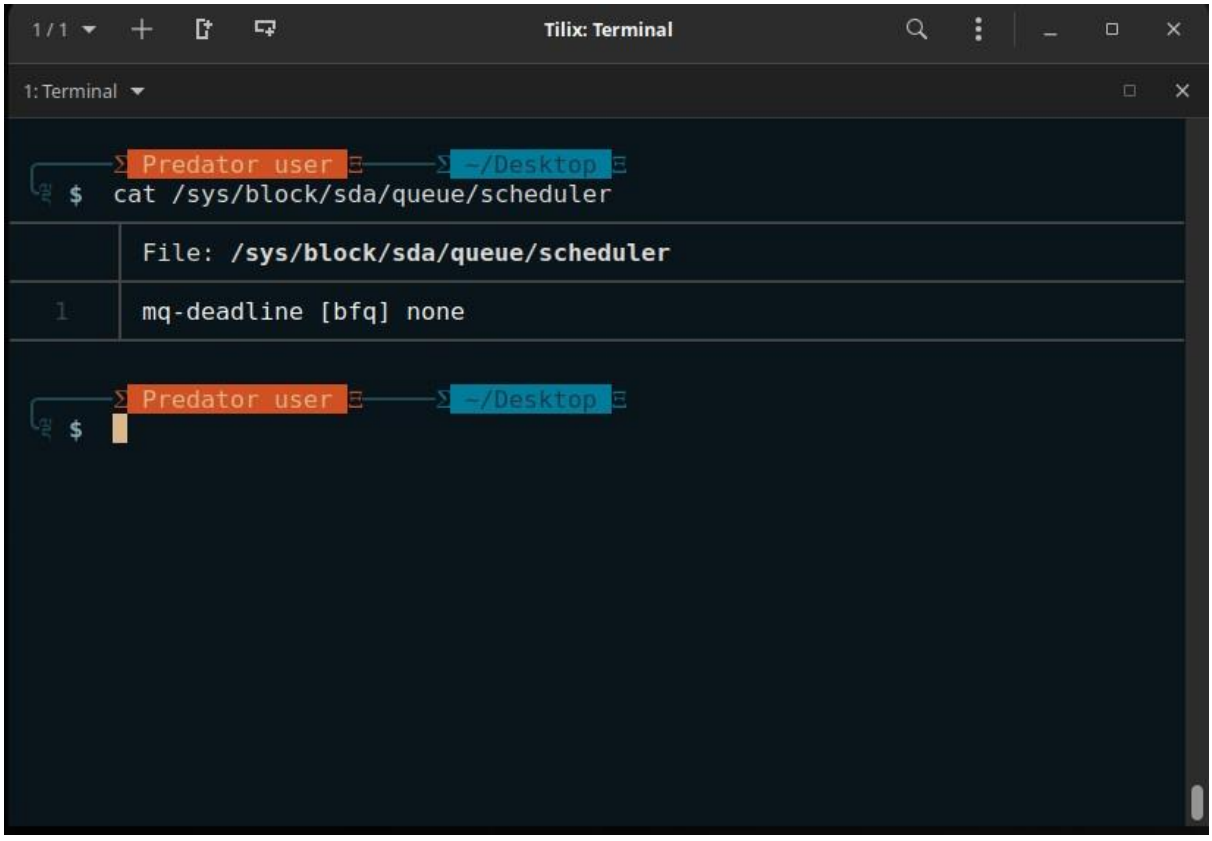
Zswap is a Linux kernel feature providing a compressed write-back cache for swapped pages, **ZRAM** creates a virtual compressed swap block in memory as alternative to a swap partition/file on disk. Both approaches increase the swapping performance and decrease the disk I/O operations.

Changing I/O scheduler

The best choice of scheduler depends on both the device and the exact nature of the workload. Also, the throughput in MB/s is not the only measure of performance: deadline or fairness deteriorate the overall throughput but may improve system responsiveness. **Benchmarking** may be useful to indicate each I/O scheduler performance.

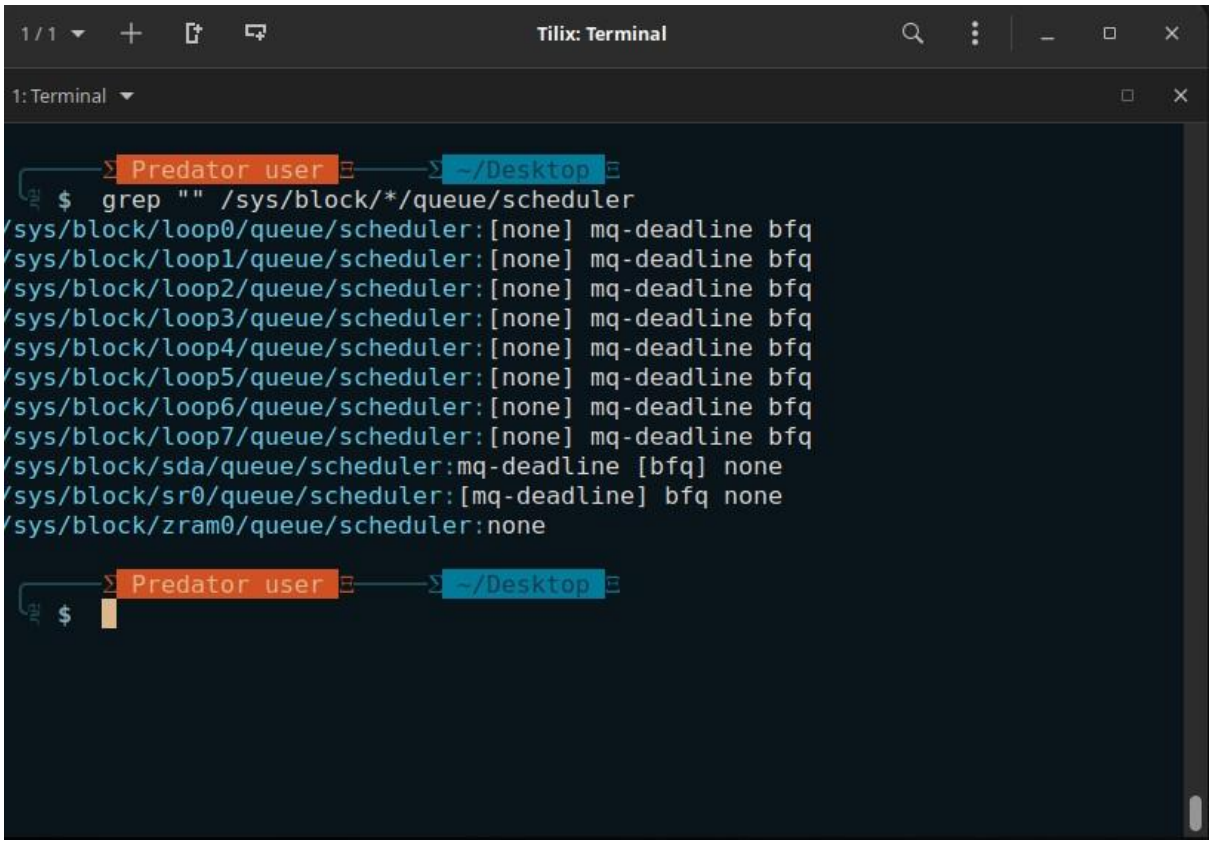
To list the available schedulers for a device and the active scheduler (in brackets):

```
$ cat /sys/block/sda/queue/scheduler mq-deadline kyber [bfq] none
```



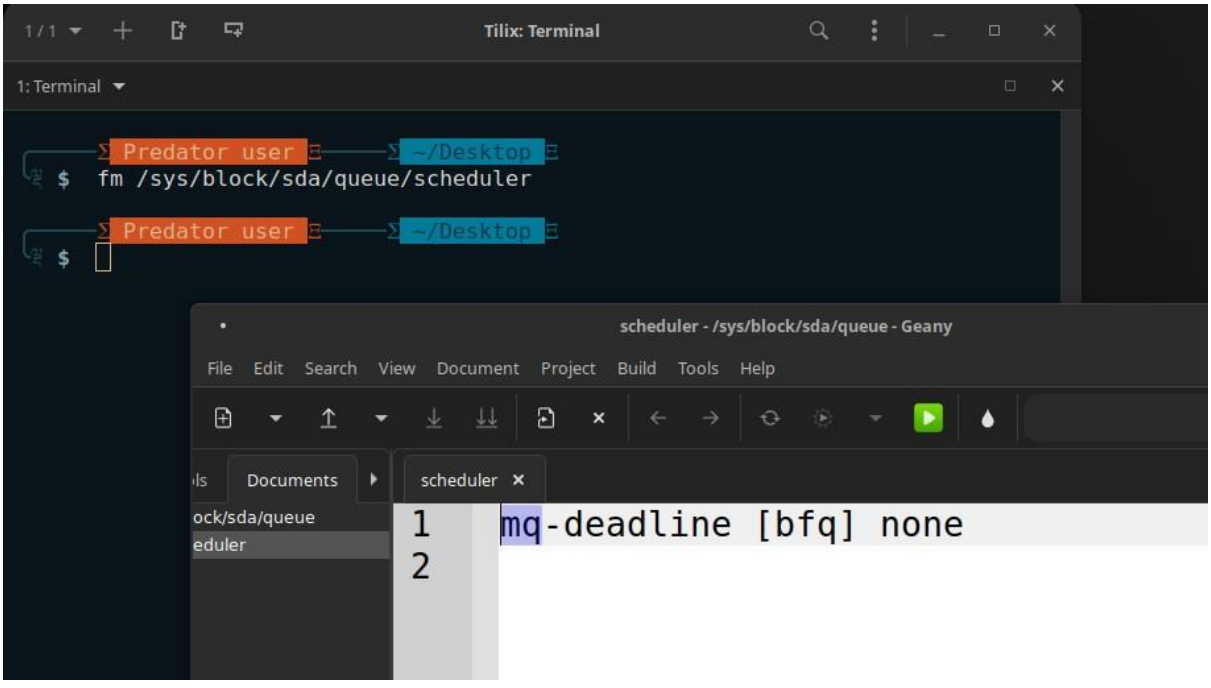
To list the available schedulers for all devices:

```
$ grep “” /sys/block/*/queue/scheduler
```



To change the active I/O scheduler to *bfq* for device *sda*, use:

```
# echo bfq > /sys/block/sda/queue/scheduler
```

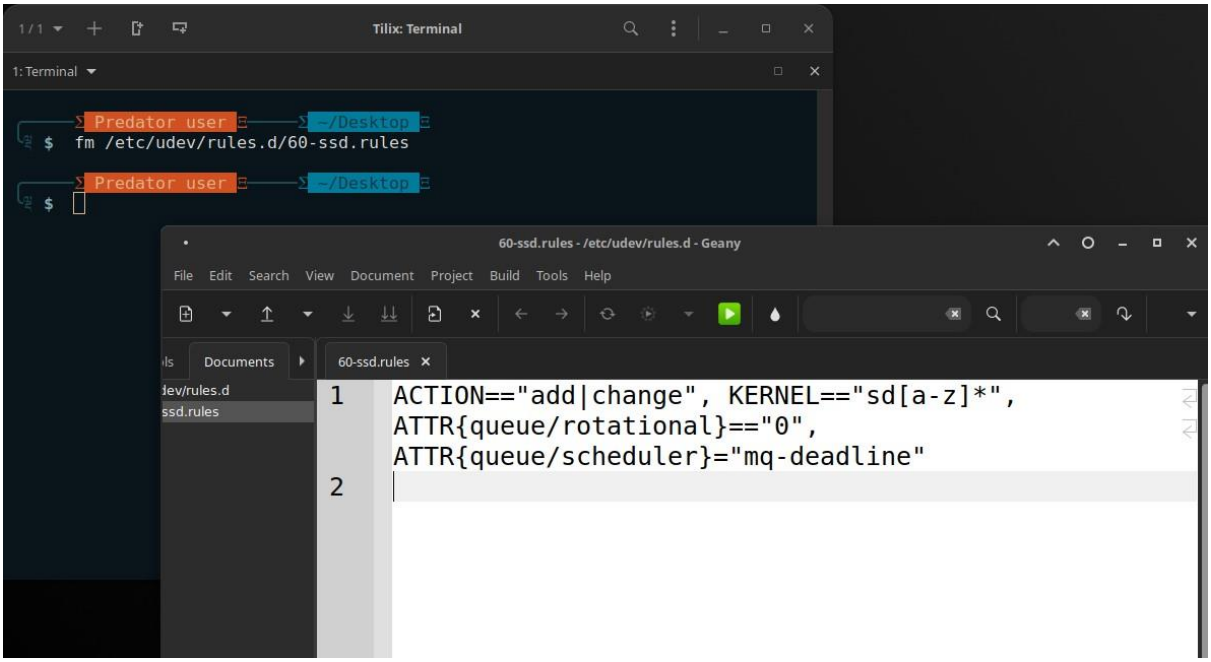


HDD I/O Scheduler Benchmarks - BFQ

If you are an SSD user, use mq-deadline IO scheduler.
If you use NVME SSD, use none IO scheduler.
To let your system select the scheduler automatically for you, use a udev rule for that!

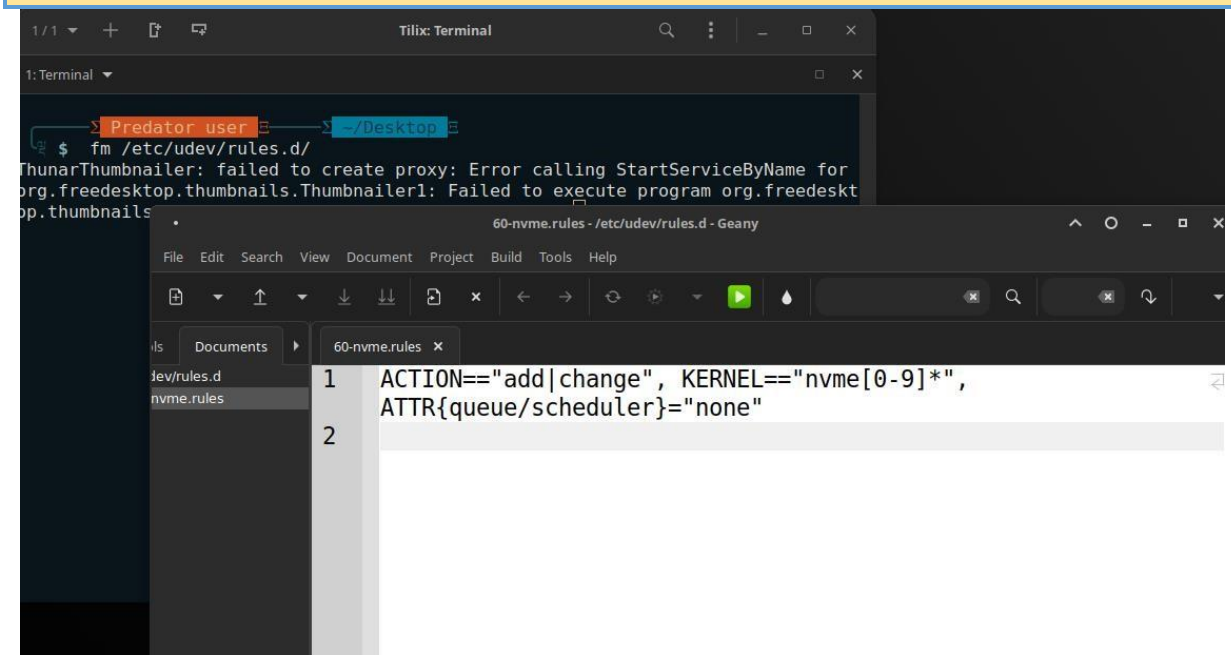
For SSDs

```
/etc/udev/rules.d/60-ssd.rules
ACTION==“add|change”, KERNEL==“sd[a-z]*”, ATTR{queue/rotational}==“0”,
ATTR{queue/scheduler}=“mq-deadline”
```



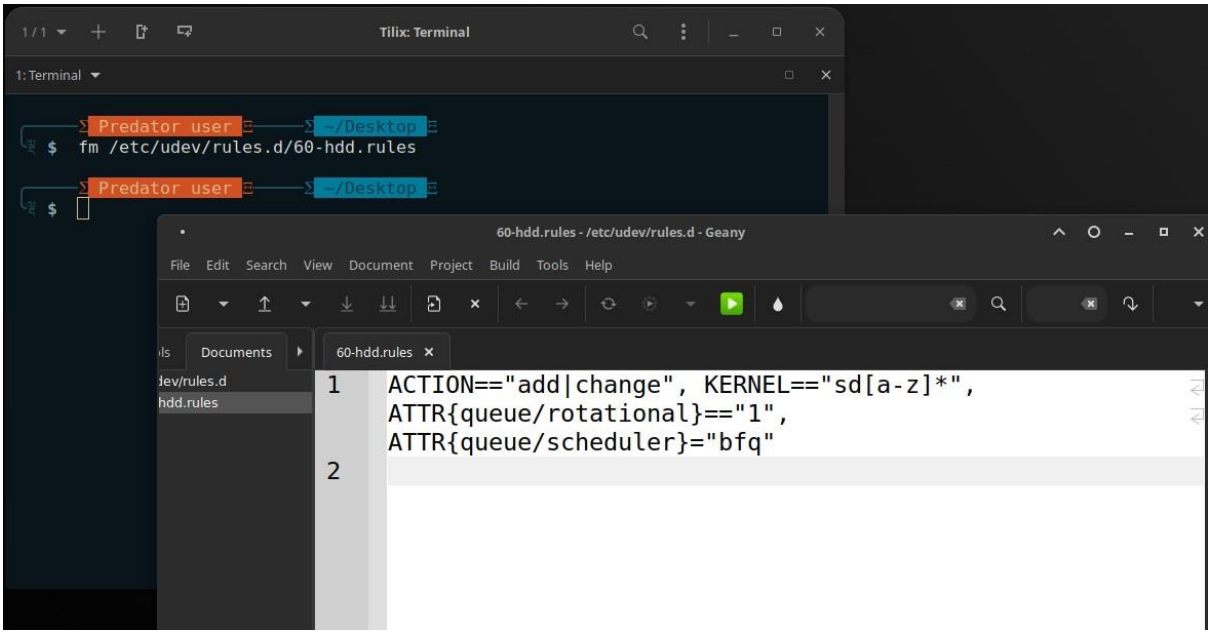
For NVME SSDs

```
/etc/udev/rules.d/60-nvme.rules
ACTION==“add|change”, KERNEL==“nvme[0-9]*”, ATTR{queue/scheduler}=“none”
```



For HDDs

```
/etc/udev/rules.d/60-hdd.rules
ACTION=="add|change", KERNEL=="sd[a-z]*",
ATTR{queue/rotational}=="1", ATTR{queue/scheduler}="bfq"
```

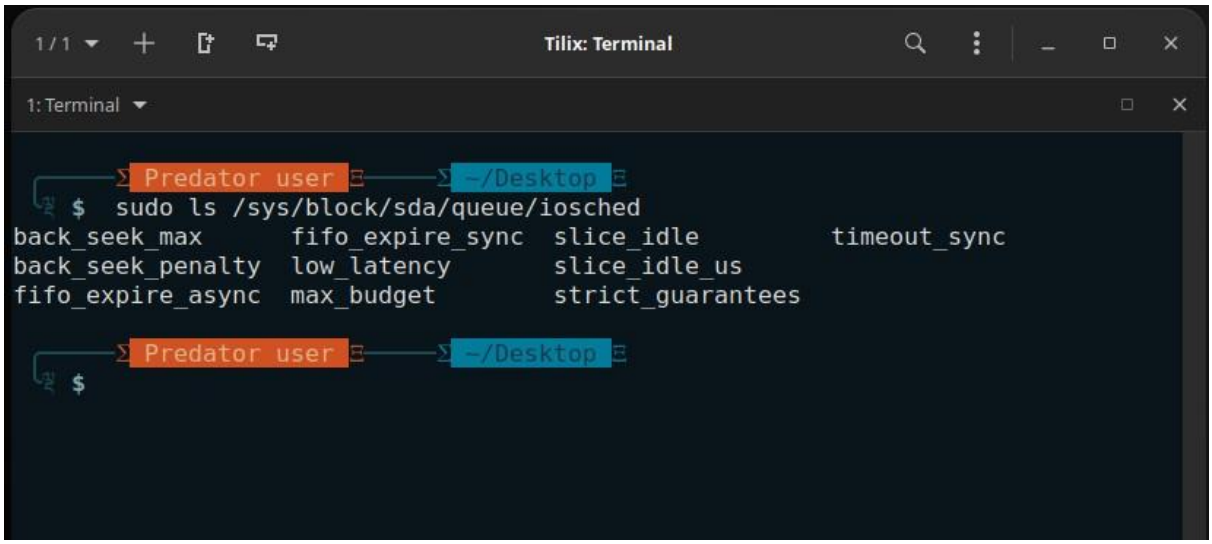


This will set the IO scheduler for all the non-rotational block devices starting from sda to sdzzz or the maximum devices supported by your system.

Tuning I/O scheduler

Each of the kernel's I/O scheduler has its own tunables, such as the latency time, the expiry time or the FIFO parameters. They are helpful in adjusting the algorithm to a particular combination of device and workload. This is typically to achieve a higher throughput or a lower latency for a given utilization. The tunables and their description can be found within the **kernel documentation**. To list the available tunables for a device, in the example below *sdb* which is using *deadline*, use:

```
$ ls /sys/block/sda/queue/iosched
```



To improve *deadline*'s throughput at the cost of latency, one can increase with `fifo_batch` the command:

```
# echo 32 > /sys/block/sda/queue/iosched/fifo_batch
```

CPU Overclocking

Overclocking improves the computational performance of the CPU by increasing its peak clock frequency. The ability to overclock depends on the combination of CPU model and motherboard model. It is most frequently done through the BIOS. Overclocking also has disadvantages and risks. It is neither recommended nor discouraged here.

Many Intel chips will not correctly report their clock frequency to acpi_cpufreq and most other utilities. This will result in excessive messages in **dmesg**, which can be avoided by unloading and blacklisting the `acpi_cpufreq` kernel module . To read their clock speed use *i7z* from the **i7z** package. To check for correct operation of an overclocked CPU

CPU performance scaling enables the operating system to scale the CPU frequency up or down in order to save power or improve performance. Scaling can be done automatically in response to system load, adjust itself in response to ACPI events, or be manually changed by user space programs.

The Linux kernel offers CPU performance scaling via the *CPUFreq* subsystem, which defines two layers of abstraction:

- **Scaling governors** implement the algorithms to compute the desired CPU frequency, potentially based off of the system's needs.
- **Scaling drivers** interact with the CPU directly, enacting the desired frequencies that the current governor is requesting.

A default scaling driver and governor are selected automatically, but userspace tools like **cpupower**, **acpid**, **Laptop Mode Tools**, or GUI tools provided for your desktop environment, may still be used for advanced configuration.

Thermald

thermald is a **Linux daemon** used to prevent the overheating of Intel CPUs. This daemon proactively controls thermal parameters using P-states, T-states, and the Intel power clamp driver. thermald can also be used for older Intel CPUs. If the latest drivers are not available, then the daemon will revert to x86 model specific registers and the Linux "cpufreq subsystem" to control system cooling.

By default, it monitors CPU temperature using available CPU digital temperature sensors and maintains CPU temperature under control, before hardware takes aggressive correction action. If there is a skin temperature sensor in thermal sysfs, then it tries to keep skin temperature under 45C.

cpupower-gui

cpupower-gui is a graphical utility designed to assist with CPU frequency scaling. The GUI is based on **GTK** and is meant to provide the same options as *cpupower*. *cpupower-gui* can change the maximum/minimum CPU frequency and governor for each core.

Setting maximum and minimum frequencies

In some cases, it may be necessary to manually set maximum and minimum frequencies.

To set the maximum clock frequency (is `clock_freq` a clock frequency with units: GHz, MHz):

```
# cpupower frequency-set -u clock_freq
```

To set the minimum clock frequency:

```
# cpupower frequency-set -d clock_freq
```

To set the CPU to run at a specified frequency:

```
# cpupower frequency-set -f clock_freq
```

Changing to acpi-cpufreq CPU management driver

The idea here is to replace the intel-pstate CPU power management driver with the acpi-cpufreq one. This allows for better performance and slightly more efficient power use in some cases, as shown here.

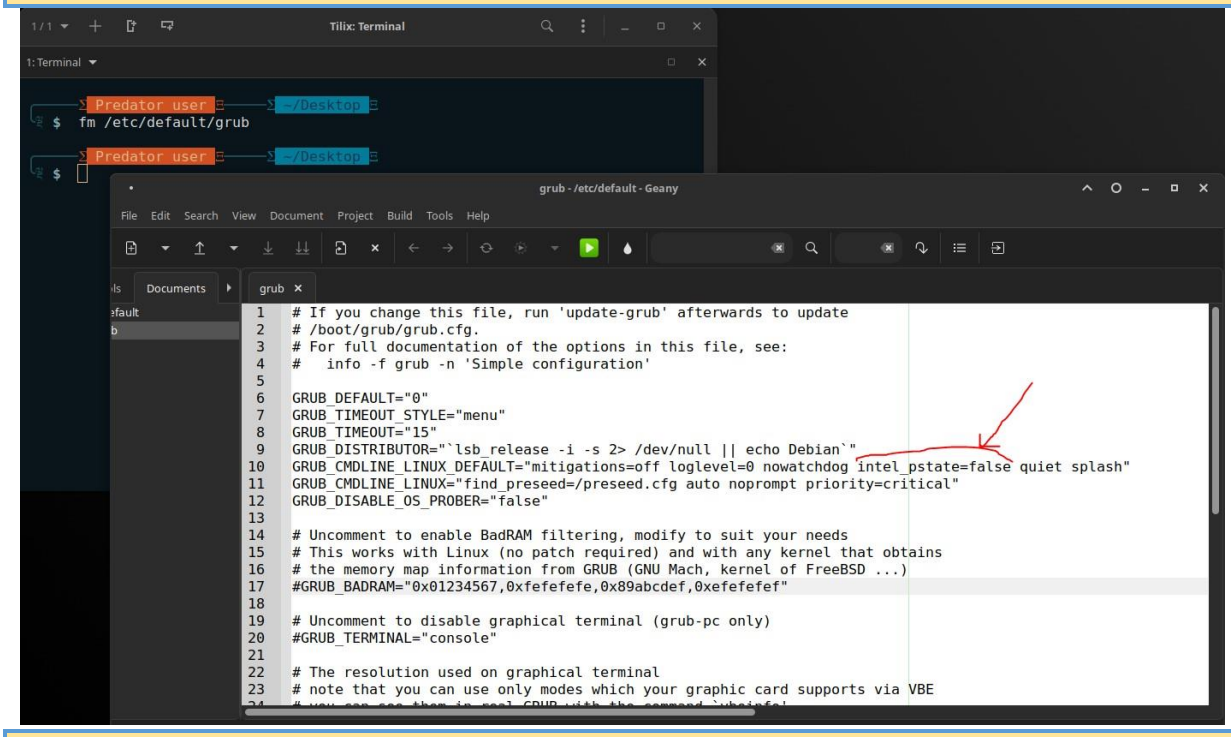
Disable intel-pstate in grub config

To disable the default intel-pstate driver, you need to edit /etc/default/grub: # also hides the splash screen for people like me that like to see log messages on boot instead of a progress bar.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet nosplash debug intel_pstate=disable"
```

After making our edits, we need to refresh grub:

```
$sudo update-grub
```




```
etc/default/grub
```

BIOS frequency limitation

Some CPU/BIOS configurations may have difficulties to scale to the maximum frequency or scale to higher frequencies at all. This is most likely caused by BIOS events telling the OS to limit the frequency resulting in `/sys/devices/system/cpu/cpu0/cpufreq/bios_limit` set to a lower value.

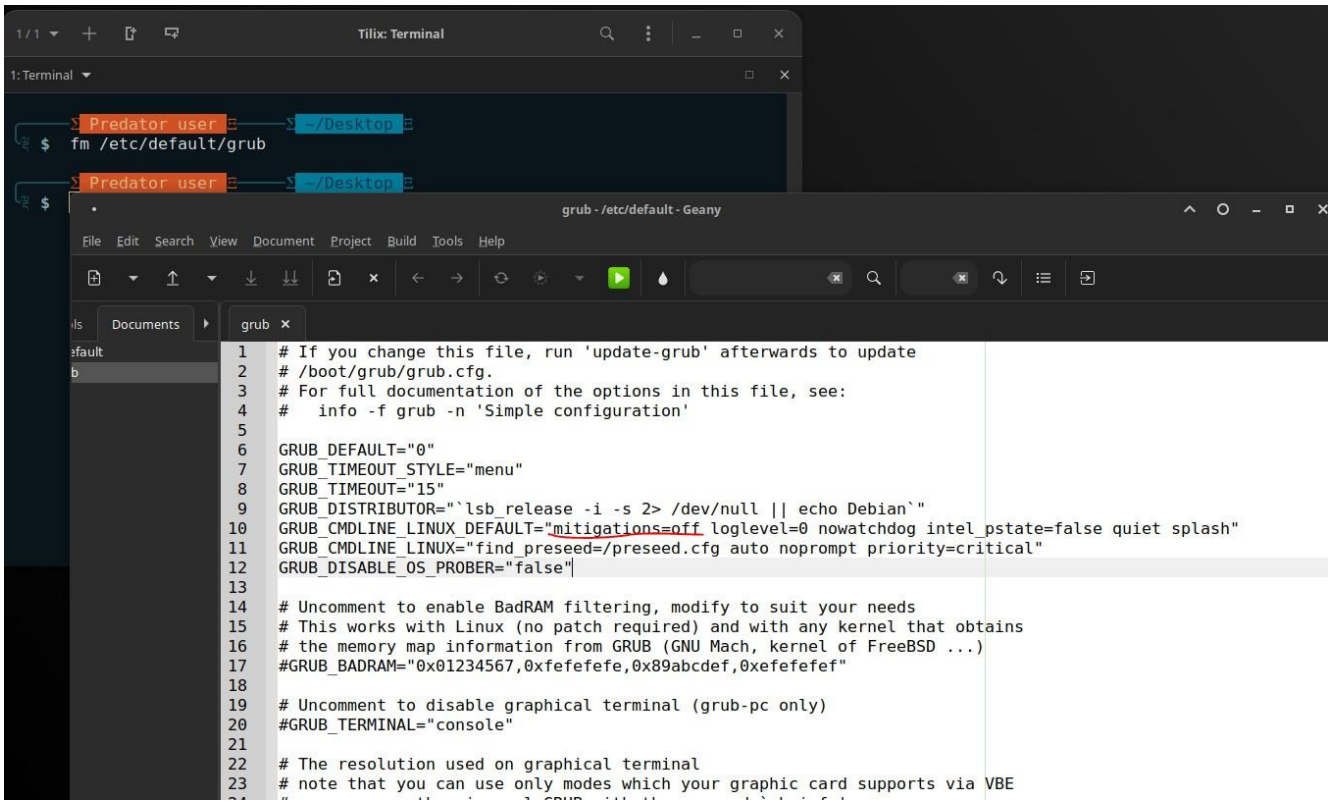
Either you just made a specific Setting in the BIOS Setup Utility, (Frequency, Thermal Management, etc.) you can blame a buggy/outdated BIOS or the BIOS might have a serious reason for throttling the CPU on its own.

Turn off CPU exploit mitigations

Warning: Do not apply this setting without considering the vulnerabilities it opens up. See **this** and **this** for more information.

Turning off CPU exploit mitigations may improve performance. Use below **kernel parameter** to disable them all:

```
mitigations=off
```



Virtual memory

There are several key parameters to tune the operation of the **virtual memory** subsystem of the Linux kernel and the write out of dirty data to disk. See the official **Linux kernel documentation** for more information. For example:

```
vm.dirty_ratio = 10
```

Contains, as a percentage of total available memory that contains free pages and reclaimable pages, the number of pages at which a process which is generating disk writes will itself start writing out dirty data.

```
vm.dirty_background_ratio = 5
```

Contains, as a percentage of total available memory that contains free pages and reclaimable pages, the number of pages at which the background kernel flusher threads will start writing out dirty data.

- Consensus is that setting `vm.dirty_ratio` to 10% of RAM is a sane value if RAM is say 1 GB (so 10% is 100 MB). But if the machine has much more RAM, say 16 GB (10% is 1.6 GB), the percentage may be out of proportion as it becomes several seconds of writeback on spinning disks. A more sane value in this case may be `3` (3% of 16 GB is approximately 491 MB).
- Similarly, setting `vm.dirty_background_ratio` to `5` may be just fine for small memory values, but again, consider and adjust accordingly for the amount of RAM on a particular system.

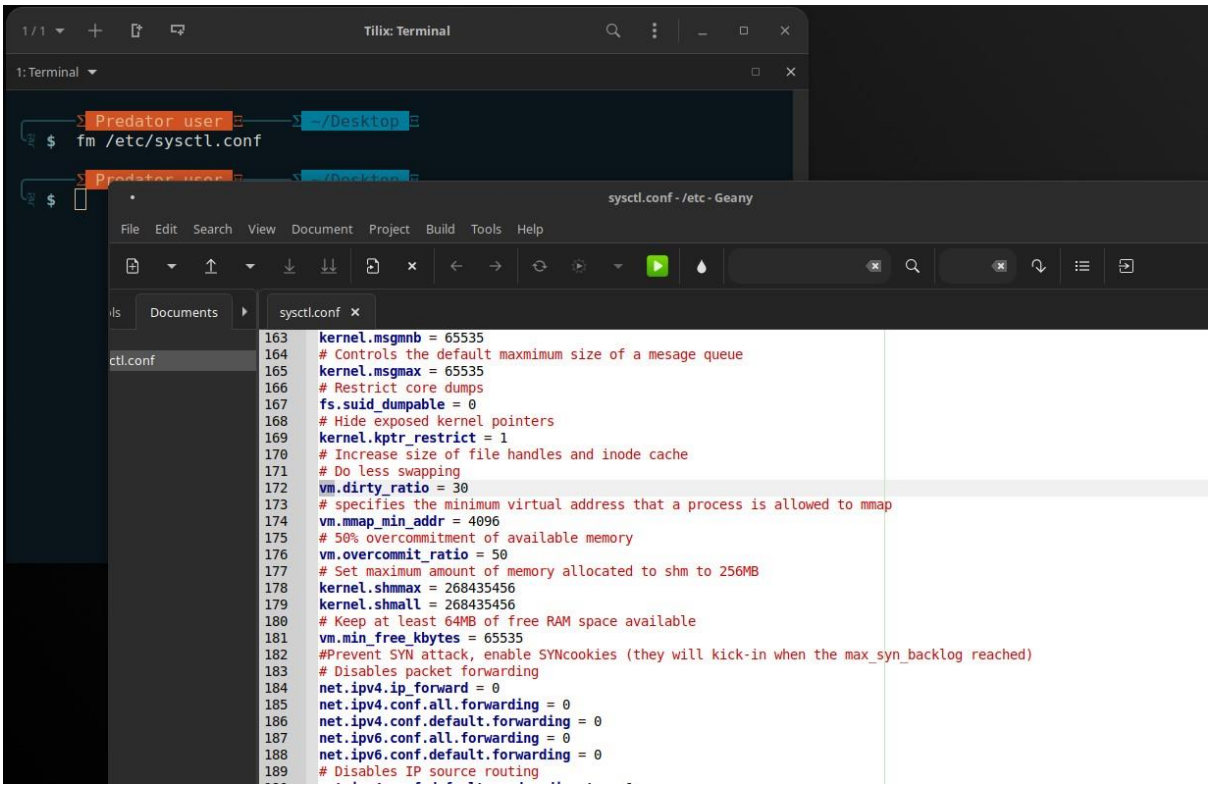
VFS cache

Decreasing the **virtual file system** (VFS) cache parameter value may improve system responsiveness:

```
vm.vfs_cache_pressure = 50
```

The value controls the tendency of the kernel to reclaim the memory which is used for caching of directory and inode objects (VFS cache). Lowering it from the default value of 100 makes the kernel less inclined to reclaim VFS cache (do not set it to 0, this may produce out-of-memory conditions).

All the configs are set in the /etc/sysctl.conf



Linux kernel configuration

You can use any of three basic methods to configure a Linux kernel. Chances are that you will have the opportunity to try all of them eventually. The methods are

- Modifying tunable (dynamic) kernel configuration parameters
- Building a kernel from scratch (by compiling it from the source code, possibly with modifications and additions)
- Loading new drivers and modules into an existing kernel on the fly These procedures are used in different situations, so learning which approaches are needed for which tasks is half the battle. Modifying tunable parameters is the easiest and most common kernel tweak, whereas building a kernel from source code is the hardest and least often required. Fortunately, all these approaches become second nature with a little practice.

Tuning Linux kernel parameters

Many modules and drivers in the kernel were designed with the knowledge that one size does not fit all. To increase flexibility, special hooks allow parameters such as an internal table's size or the kernel's behavior in a particular circumstance to be adjusted on the fly by the system administrator. These hooks are accessible through an extensive kernel-to-userland interface represented by files in the **/proc** filesystem (aka procfs). In some cases, a large user-level application (especially an infrastructure application such as a database) might require a sysadmin to adjust kernel parameters to accommodate its needs.

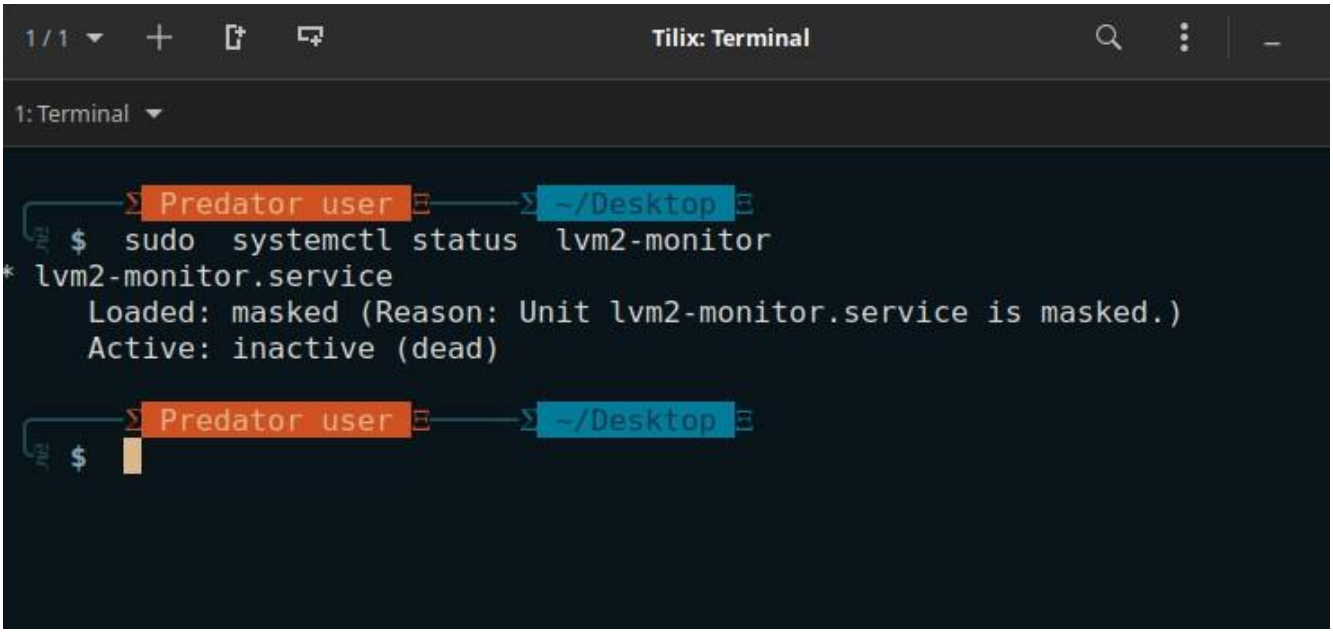
To avoid duplicate in sysctl.conf

```
#cat /etc/sysctl.conf|grep -v ^#|grep -v ^$|awk -F"=" '{print $1}'|uniq -c|awk '{print $1}'|grep -v 1
```

Logical Volume Manager (LVM)

it is a device mapper framework that provides logical volume management for the Linux kernel. Most modern Linux distributions are LVM-aware to the point of being able to have their root file systems on a logical volume. The lvm monitor service is disabled by default.

```
# systemctl status lvm2-monitor
```



Predator-OS sollution was to reduce the timeout, and to do so without messing with the base installed networking.services systemd file. This will persist during updates in any package.

hugepages in /etc/sysctl.conf file.

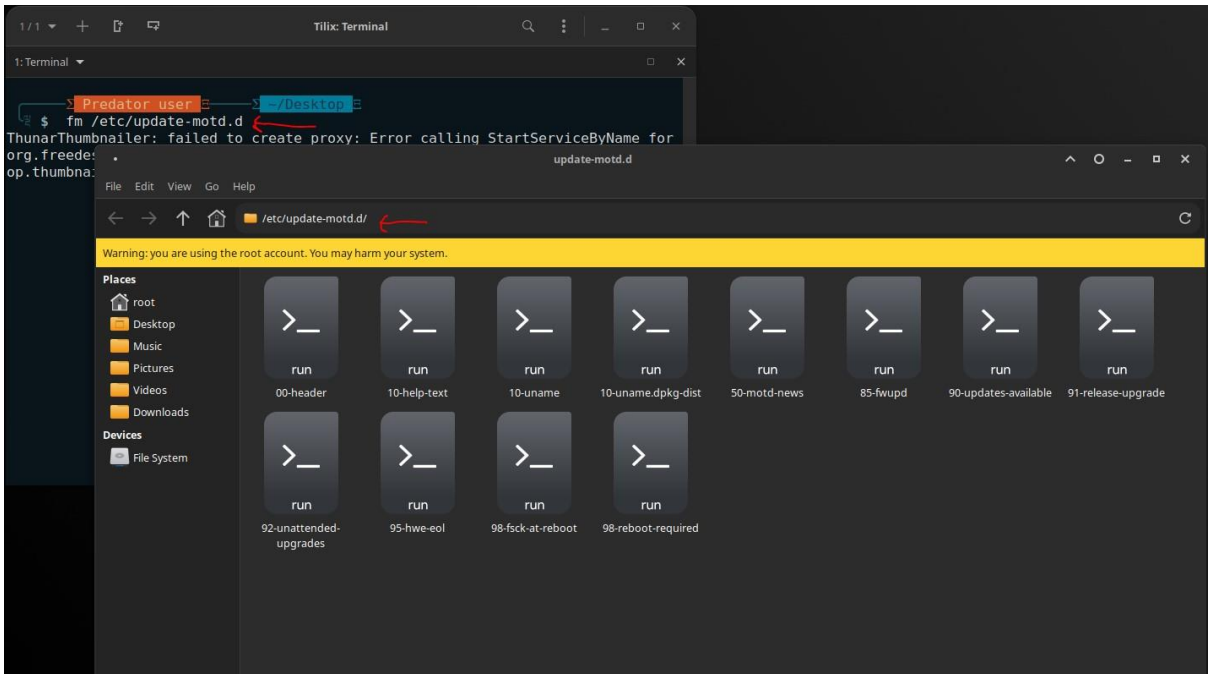
```
vm.nr_hugepages = 126 126 pages x 2 MB = 252 MB
```

```
$cat /sys/kernel/mm/transparent_hugepage/
```

Motd

motd - message of the day

The welcome message shown to a user upon the terminal login whether it is via remote SSH login or directly via TTY or terminal is a part of **motd** also known as "**M**essage **O**f **T**he **D**ay" daemon. The **motd** message can by customized to fit individual needs of each user or administrator by modifying the **/etc/motd** file or script within the **/etc/update-motd.d** directory.



Modifying the **/etc/motd** file is fast and effective way on how to quickly change the welcome message. However, for more elaborate configuration it is recommend to customize the MOTD via scripts located within the **/etc/update-motd.d** directory.

```
/etc/default/motd-news
/var/cache/motd-news
/etc/update-motd.d/*
```

Configuration Directories

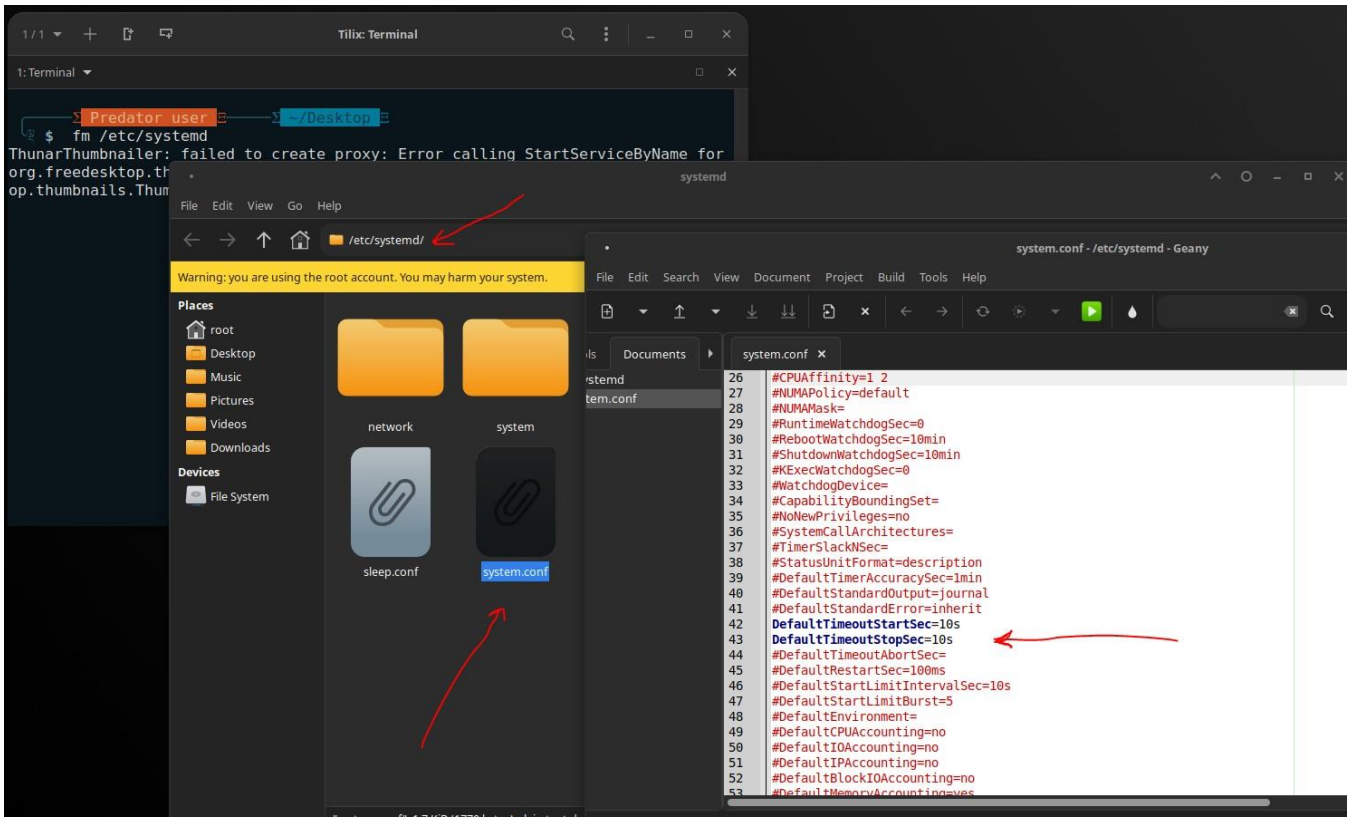
The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration

file in /etc/systemd/ contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from /usr/lib/systemd/*.conf.d/, /usr/local/lib/systemd/*.conf.d/, and /etc/systemd/*.conf.d/. Those drop-ins have higher precedence and override the main configuration file. Files in the *.conf.d/ configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under /usr/. Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in the configuration directory in /etc/, with the same filename as the vendor configuration file.



/etc/systemd/system.conf

DefaultTimeoutStartSec=10s DefaultTimeoutStopSec=7s
Then:
$sudo systemctl daemon-reload

OS Prober

What is OS Prober? What is issue with new release?

If you are multi-booting with other Linuxes, and Windows, you might find an issue, when you update or upgrade Debian stable (maybe with other Linuxes too) sometime now, it'd stop "seeing" other distros and Windows. The issue here is in the GRUB 2.06 it is disabled for OS-detecting feature security.

What file to edit?

You need to edit the Grub configuration file which is located on:

```
sudo nano /etc/default/grub
```

Make sure that you are a root or root privileged user who can edit.

To disable the OS Prober, use the following command.

```
GRUB_DISABLE_OS_PROBER=true
```

To enable the OS Prober, use the following command.

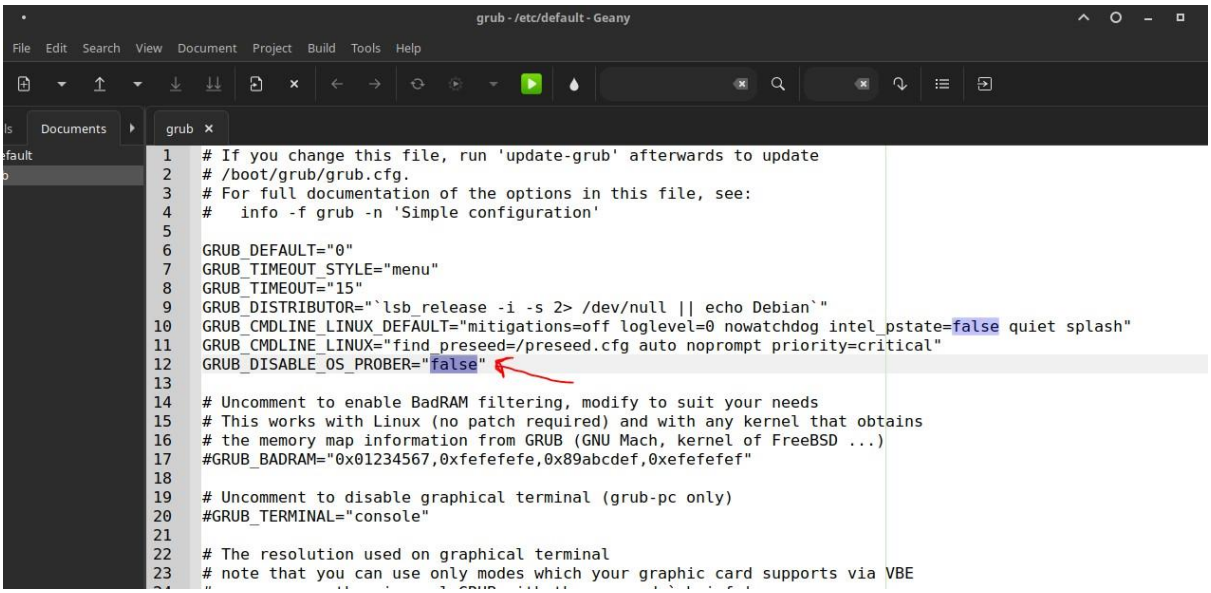
```
GRUB_DISABLE_OS_PROBER=false
```

Once you have set the instruction you can save and exit the file.

Now just update the Grub so it can take effects.

```
sudo update-grub
```

This is it, just reboot (restart) the system and check it.



GRUB_DISABLE_OS_PROBER=false

Secure Boot

To check whether your system has Secure Boot enabled or disabled, type: /usr/bin/mokutil --sb-state

/usr/bin/mokutil --disable-validation


1 / 1
Tilix: Terminal
1: Terminal
Predator user
~/Desktop
$ /usr/bin/mokutil --disable-validation
password length: 8~16
input password:

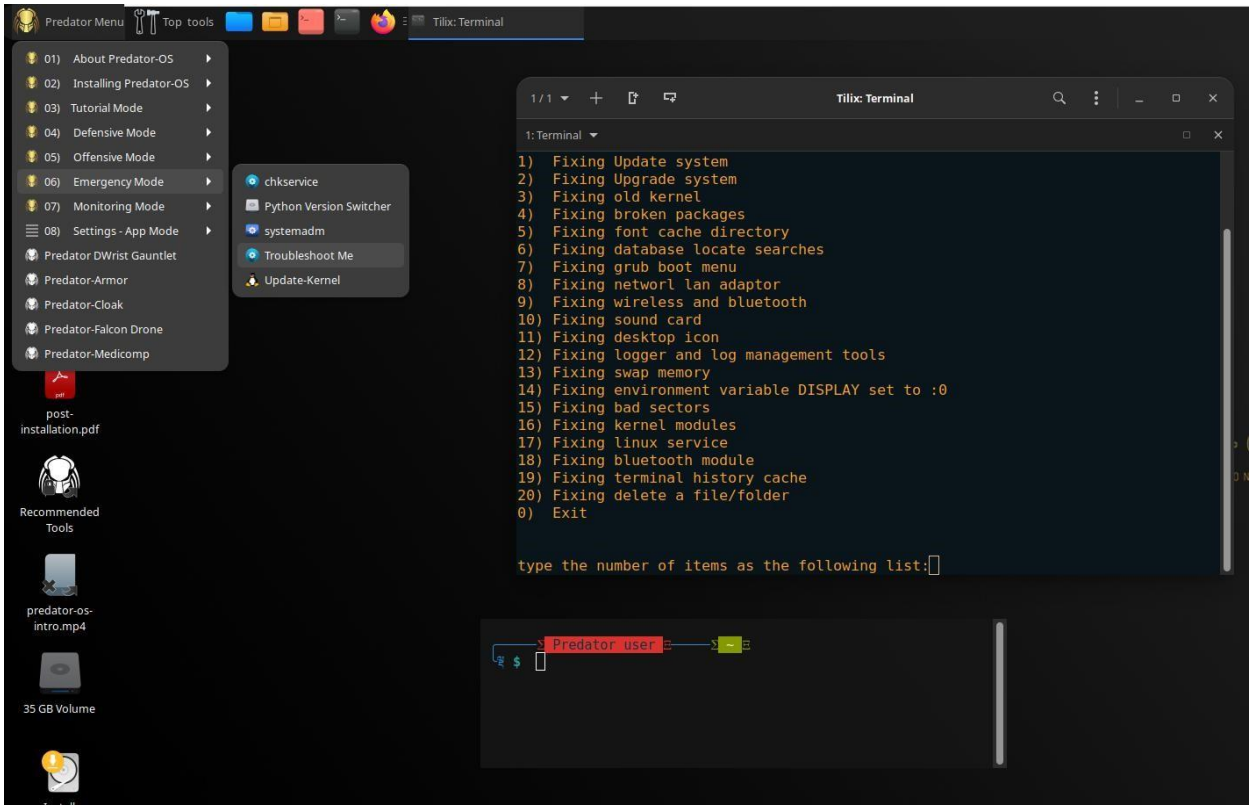
How to Bug Report

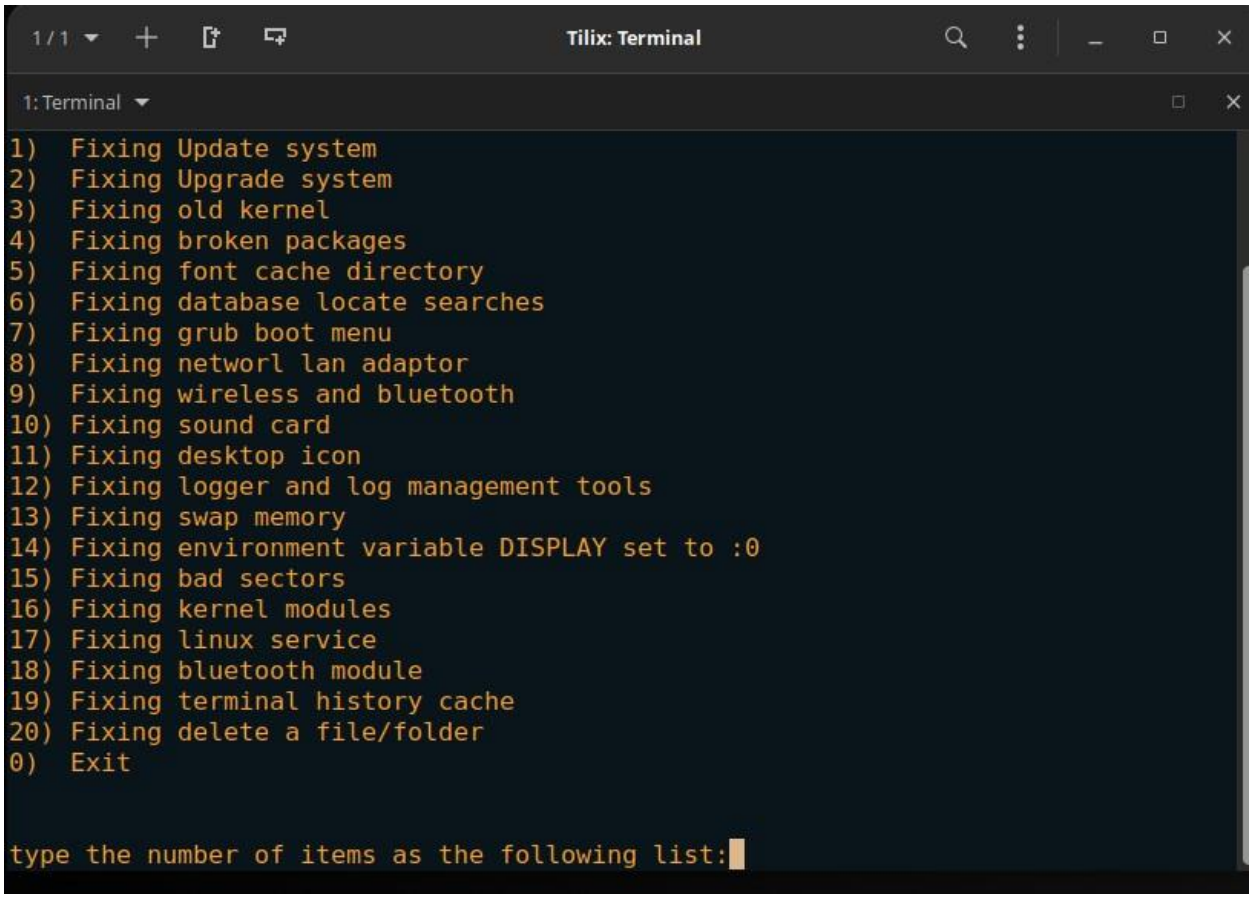
There is issues on github page to track the bugs or send an issue of Predator-OS.

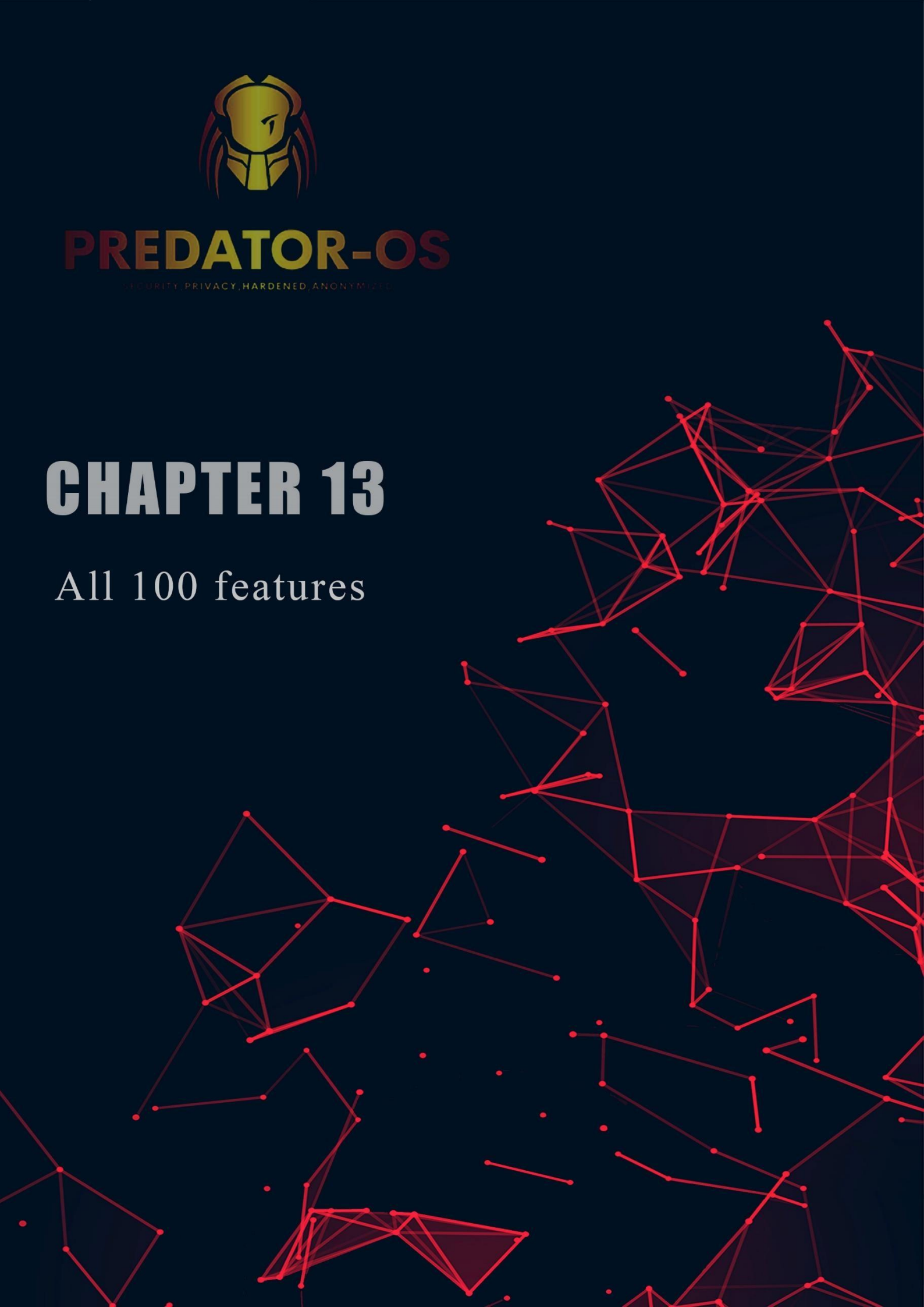
Please write any bug in the following link:

https://github.com/hosseinseilani/Predator-OS/issues/

There is a sctrip on emergency mode to troubleshooting your system.







CHAPTER 13

All 100 features

Chapter 13

All 100 features

All 100 Features

Features 1-10

1.security-oriented Linux distribution

-.-

easy-to-install and pre-configured system featuring

2.Plasma as the default desktop environment

Fully Customization Desktop and menu

3.Operates at 9 several differnt modes

Privacy, Secure, Hardened, Anonymized, Tutorial, Defensive, Offensive, Monitoring, Emergency

4.Update Manager Tools

----

kernel and tools update manager, Update notifications can notify the user if their system is becoming out of date

5.collection of security tools
Predator Linux has around 1300 pre-installed tools which are split into 40 categories

6.Graphical boot system and logger

OS has a plymouth and you can press arrow key to see the console messege

7.Live Mode by default, Live DVD Live USB

Also, You can run the installer and install it

8.Plasma-Panel

Application launchers, menus, a workspace switcher and more plugins

9.Ulauncher

It very easy to find and launch your files, quickly open websites, find programs to open, calculate math problems, and more

10.Customized Menu

Os Allows easy and fast access to all pre-installed app, organised into 40 categories

Features 11-20

11.Terminal in Desktop Background

---

highly configureable Graphical, Always be open and running in desktop background

12.disabled suspend, Hibernate, sleep

Because of prevent access to the entire memory dump and private data, passwords, keys security threat

13.Privacy Tools

PrivacyTools provides services, tools and knowledge to protect your privacy against global mass surveillance.

14.fully OSINT Framword

Focused on gathering information from free tools or resources

15.Bug Bounty Tools

Give users the ability to harness a large group of hackers in order to find bugs in their code.

16.Included Cyber Search Engine

Private Search Engines with popular features and latest links

17.CTF Tools

Running a capture the flag (CTF) competition, Tools used for solving CTF challenges

18.Penetration Testing Labrator

There are many tools to learn the practical side of vulnerability assessments and virtual space to test malwares

19.Most Scaner Drivers

More than 2500 scaner devices supported by sane package

20.Privacy-enhanced browsers

---

privacy focused browser,shields FOR privacy,Blocking harmful ads

Features 21-30

21.Secure Group Chat Tools

---

included secure messaging app. End-to-end encryption, multi-mode messaging,Multi-platform support

22.self-destruct Tool

Clean Every Logs, tracks,memory,cache and fast Shutdown PC security threat

23.VPN Tools

---

secure all data communications and extend private network.securely access corporate network through an encrypted connection

24.Smart and user-friendly SHELL

Included zsh and fish shell

25.Included IDS/IPS Tools

Antivirus,Trojan Remover, malware Detection, apparmore,AIDE, SElinux and Firewalls

26.Integrity Check Tools

Included tools to compare state of stored data and file integrity monitoring

27.Apparmor Profiles

Installed and enabled all available apparmor profiles

28.Included USB Guard Tools

Prevent from USB, physical access and Electrical attacks

29.Included Tune utility

The tuned command-line tool allows users to switch between different tuning profiles.

30.Real time monitoring Tool

Monitoring and troubleshooting everything in real time for free with Netdata Features 31-40

31.Google Hacking Database

You can access to more than 6 entries

32.controling All services

---

included Stacer tool to controling all services beside the systemctl cli

33.Removing Unnecessary data

Removing Unnecessary services, packages, startup files and configs

34.disabled the [Ctrl]-[Alt]-[Delete]

Disallow anyone to reboot the OS using Ctrl-Alt-Del keys.

35.Included Database Tools

---------- included

the most useful database management tools

36.Included Audit Tools

Included Security Audit and Intrusion Detection Tool: tiger,lynis and more tools

37.included Mix Networks protocols

Hard-to-trace communications, by encrypting and keep you anonymous online.such as i2p,Tor,OpenVPN,Lantern

38.Included osquery

Allows you to craft your system queries using SQL statements to troubleshoot performance and operational issues

39.Included log management tools

Managing and analyzing log files, record system activity, events and report generation tools

40.password lists

Access to 300 GB password list contains every wordlist, dictionary, and password database leak

Features 41-50

41.included wipe tools

---

fully erase data from hard drive, memory, rewrites the sector and flushes the cache

42.Included Virtual Keyboard

Florence is an open source extensible scalable virtual keyboard

43.Malware Analysis Toolkit

---

included Toolkit for reverse-engineering and analyzing Windows and Linux malicious,Forensic investigators

44.Machine Learning for Cybersecurity

---

included python2 and 3 modules for InfoSec, malware analysis,cyber security and Machine Learning

45.Self-document of tools

Describes the application with extra lines as a tools tooltip.

46. package managers

Included apt, aptitude, alien, synaptic, dpkg, snap

47.Kernel tuning

---

included Kernel tuning configurations such as high cache pressure,tuned performance profile

48.Included cpupower Tools

Included collection of tools to examine and tune power saving related features of your processor.

49.Data Recovery Tools

Included Tools for lose data, critical informationeither through accidental deletion, virus attacks,corrupted data permanent removal of files

50.Forensic Metapackage

All here available tools are packaged by Debian Security Tools Team

Features 51-60

51.included Sleuth Kit

---

collection of command line tools to analyze disk images and recover files from them

52.ExecShield Buffer Overflows

Protecting against buffer overflows

53.Securing network access

Protected against syn packet flooding and ARP attacks

54.Disabled Auto-mounting

Turn off removable drives Auto-mounting feature

55.Disabled virtual machine shared folders

Disabled Mounting virtual machine shared folders between host and guest

56.Normal user as defualt

Used normal user as defualt,Consider disabling the root account permanently and run the command with sudo

57.Volume or container encryption

Included symmetric and asymmetric encryption tools for files and volumes

58.Included ASLR

Address Space Layout Randomization.ull randomization prevent the exploitation of memory corruption vulnerabilities

59.x86 compatible and Virtual Machines

Supported x86 compatible and Virtual Machines and virtualization

60.Forensic Metapackage

All here available tools are packaged by Debian Security Tools Team

Features 61-70

61.Protecting Network Traffic

Included a collection of Tor, torify, and torsocks, ProxyChains to keeping OS anonymized

62.Protection against location discovery

Protection against IP address,location discovery,Geolocation, or location services,,

63.MAC address Changer

Protected against spoofing of the MAC address, generate a random value for each connection or manually by custome tool

64.Cold boot attack protection

Prevents attacks by disabled hibernation and sleep modes, wipe memory,full-disk encryption

65.Prevent NTP amplification attack

Disabled Network time synchronization which synchronize system clock between devices to prevent NTP amplification attack.

66.Updated Microcode

Applied CPU microcode updates

67.Password Manager Tools

Included Trusted password managers,manage your passwords in a secure way

68.Disabled Microphones

Disabling Microphones by defualt

69.Metadata Cleaner

Included Metadata Removal Tools

70.VirtualBox Hardening

VirtualBox Hardening

Features 71-80

71.Disable ACPI

Disable Advanced Configuration and Power Interface (ACPI)

72.Restricting kernel modules

Restricting module loading can protect the kernel.

73.Disabled bug reporter

Disabled bug reporter software and services

74.preventing spoofing attacks

---

preventing spoofing attacks

75.Enable TCP/IP SYN cookies

Enable TCP SYN cookie protection to save domain from SYN Attack

76.Packet forwarding for IPv4/v6

Permits the kernel to forward packets from one network interface to another

77.Prevent MITM Attacks

Do not accept ICMP redirects to prevent MITM attacks

78.Accept ICMP redirects

Accept ICMP redirects only for gateways listed in our default

79.Size of inode cache

Increase size of file handles and inode cache

80.Swapping and Caching

Increase size of swappiness and cache pressure Features 81-90

81.Dirty Rratio

Better Linux Disk Caching & Performance

82.Included Network security option

Increasing times SYNACKs for passive TCP connection

83.Protect Against TCP Time-Wait

Protect Against TCP Time-Wait

84.Control Syncookies

Control Syncookies

85.Socket Send/Receive Buffer

Increasing Socket Receive Buffer

86.Increase memory buffers

Increase the maximum read/write of option memory buffers space

87.Prevent DOS attacks

Increase the tcp-time-wait buckets pool size to prevent simple DOS attacks

88.Increasing pid number

Increasing pid number

89.Disabled IPv6

Disabled IPv6

90.Enable IP spoofing protection

Enable IP spoofing protection

Features 91-100

91.kernel.randomize_va_space

Random positions in a process's address space, which makes it difficult for an attacking program to predict the memory address of the next instruction

92. Tuning TCP buffer

Increase Linux auto tuning TCP buffer limits

93.Prevent Brute force attacks

---

preventing brute force attacks, consider implementing the intrusion prevention software Fail2ban

94.Enabled SELinux

Enforcing mode is the default and enforcing the loaded security policy on the entire system.

95.Disable IP source routing

Accept packets with the Strict Source Route (SSR) or Loose Source Routing (LSR)

96.protects against time-wait assassination

---

protects against time-wait assassination by dropping RST packets for sockets in the time-wait state.

97.prevent man-in-the-middle attacks

---

disable ICMP redirect acceptance and sending to prevent man-in-the-middle attacks and minimise information disclosure.

98.avoid Smurf attacks

Ignore all ICMP requests to avoid Smurf attacks, make the device more difficult to enumerate on the network and prevent clock fingerprinting through ICMP timestamps.

99.Wireless devices blacklisted

---

rfkill to reduce remote attack surface further. To blacklist all wireless devices

100.Grsecurity

It will be add in next update

Social medias

http://t.me/UNIDENTIFIED_TM http://t.me/predator_os
https://www.linkedin.com/in/hossein-seilany-2931891b4
https://github.com/hosseinseilani/

Telegram
@seilany

GitHub
https://github.com/hosseinseilani/

Youtube
https://www.youtube.com/@predator-os5453

Email
info.predator-os@gmail.com

Linkedin
https://www.linkedin.com/in/hossein-seilani

pinterest

https://www.pinterest.com/hosseinseilanii/

All about me **https://seilany.ir/**

These are another distro that I developed.

Emperor-OS

It has more than 500 applications in 20 categories for programming, graphic design, and data science.Comes in 64-bit ISO and has 10 desktops.

see Website

V2.5

Little-Psycho

It is lives CD with a KDE plasma.t is a Dangerous, Wild, Destructive, changer, stress testing, system, and resilience testing Linux for hardware and software.

see Website

V 1.0

Hubuntu

It is hardened Ubuntu that secure for sensisensitive desktop user and environments.Based on Ubuntu 18-20-22-24 LTS.

see Website

LTS

Artystone

It is a desktop distro especially highly customized for general usage.Based on Debian,Ubuntu and Arch.with more features.

see Website

Founder and Developer

I am Hossein Seilani, M.S. in Computer Science, and the founder and developer of Emperor-OS, Little-Psycho, and Predator-OS Linux. I have experience and certifications in various domains, including Linux/Windows Sysadmin, UX/UI, Front-End web design, SEO, Graphic Designer, Data Science, and machine learning.

ALL ABOUT ME