

Linux Core: Kernel and Filesystem Architecture

Check GitHub for helpful DevOps tools:

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>




Ask Personal AI Document assistant to learn interactively (FASTER)!

1

Download PDF

1

<https://github.com/MichaelRobotics/DevOpsTools/blob/main/LinuxCore.pdf>

 | Click there to go to ChatPdf website



2

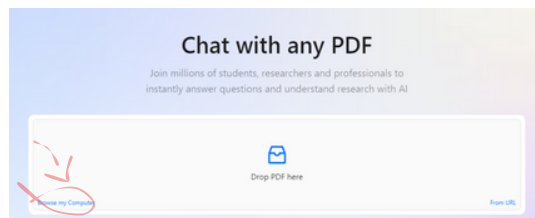
2

Go to website

3

Browse file

3



4

Chat with Document

Ask questions about document!

4

Completly new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

 <https://www.hackthebox.com/>



What is Linux Core?

The Linux core, also known as the kernel, is the central part of the Linux operating system that manages hardware resources, memory, and processes. It acts as a bridge between software applications and the underlying hardware.

How do kernel and filesystem works?

The kernel handles communication between hardware and software by managing tasks such as process scheduling, memory allocation, and device control. The filesystem provides a structured way to store, organize, and access files on storage devices, allowing the kernel to read and write data efficiently.

Kernel & Filesystem: Why and When it is usefull to know

Understanding the kernel and filesystem is essential for troubleshooting performance issues, optimizing system behavior, and developing or configuring software that interacts directly with the OS.

This knowledge is especially valuable when working on system administration, embedded systems, or software development.

System Requirements

- 1 GB RAM (minimum, 2 GB recommended for better performance)
- 10 GB free storage (more may be required depending on the size of the DNS zone files and logs)
- Ubuntu 22.04

Linux kernel: Main components & packages

linux-image – The core kernel package that includes the binary kernel image, enabling the operating system to boot and manage hardware resources.

linux-modules – A package that provides various kernel modules (drivers) to support a wide range of hardware devices and peripherals, which can be loaded and unloaded as needed.

linux-headers – Contains the header files necessary for compiling additional modules, custom kernels, and software that interacts directly with the kernel.

linux-firmware – Provides firmware binaries needed for hardware devices such as network cards, GPUs, and wireless adapters, ensuring proper functionality.

Linux CORE: Filesystem

1) Filesystem directories and their purpose

First tool that will help you understand Linux filesystem structure is: tree

```
sudo apt install tree
```

Now execute:

```
tree -L 1 /
```

Your root filesystem should be shown similar or as follows:

```
Laptopdev@laptopdev2:~$ tree -L 1 /
/
├── bin -> usr/bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib32 -> usr/lib32
├── lib64 -> usr/lib64
├── libx32 -> usr/libx32
├── lost+found
├── media
├── mnt
├── name
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── snap
├── srv
├── swapfile
├── sys
├── tmp
├── usr
└── var
```

Now lets dive deeper into explanation of purpose of each directory.

/bin

Directory containing essential binaries, including core applications and programs like ls, cp, mv, and rm. These tools are used for basic file management tasks, and other /bin directories exist throughout the filesystem for specific purposes.

/boot

Contains essential files for booting the system. Altering these files can prevent Linux from starting, but superuser privileges are required for any changes.

/dev

Holds device files, which are created at boot or dynamically when new hardware, like USB drives or webcams, is connected.

/etc

Originally a "miscellaneous" directory, it now stores system-wide configuration files, including settings for system name, users, passwords, network, and disk partitions.

/home

Contains personal directories for each user, such as /home/paul for user files and /home/guest for temporary access.

/lib

Contains libraries with code for applications, including essential kernel modules that enable hardware functionality like video, audio, and network drivers.

/media

The directory where external storage devices, like USB drives and SD cards, are automatically mounted when connected.

/mnt

A legacy directory for manually mounting storage devices or partitions, now used infrequently.

/opt

Directory for software compiled from source, with applications in /opt/bin and libraries in /opt/lib. Another common location for such software is /usr/local, which also has its own bin and lib directories.

proc

A virtual directory containing real-time information about the system, including CPU details and kernel status, generated dynamically at runtime.

/root

The home directory for the superuser (Administrator), kept separate to prevent regular users from modifying it.

/run

A directory for system processes to store temporary data; it's not meant for user access.

/sbin

Similar to /bin, this directory holds applications for the superuser, often requiring sudo for execution; it includes critical tools for system maintenance and management.

/usr

Originally for user home directories, this directory now contains shared applications, libraries, documentation, and other resources. It includes its own bin, sbin, and lib directories, with many modern distributions linking /bin and /usr/bin for simplicity.

/srv

Contains server data; for example, web server HTML files are stored in /srv/http, while FTP server files go in /srv/ftp.

/srv

Contains server data; for example, web server HTML files are stored in /srv/http, while FTP server files go in /srv/ftp.

/tmp

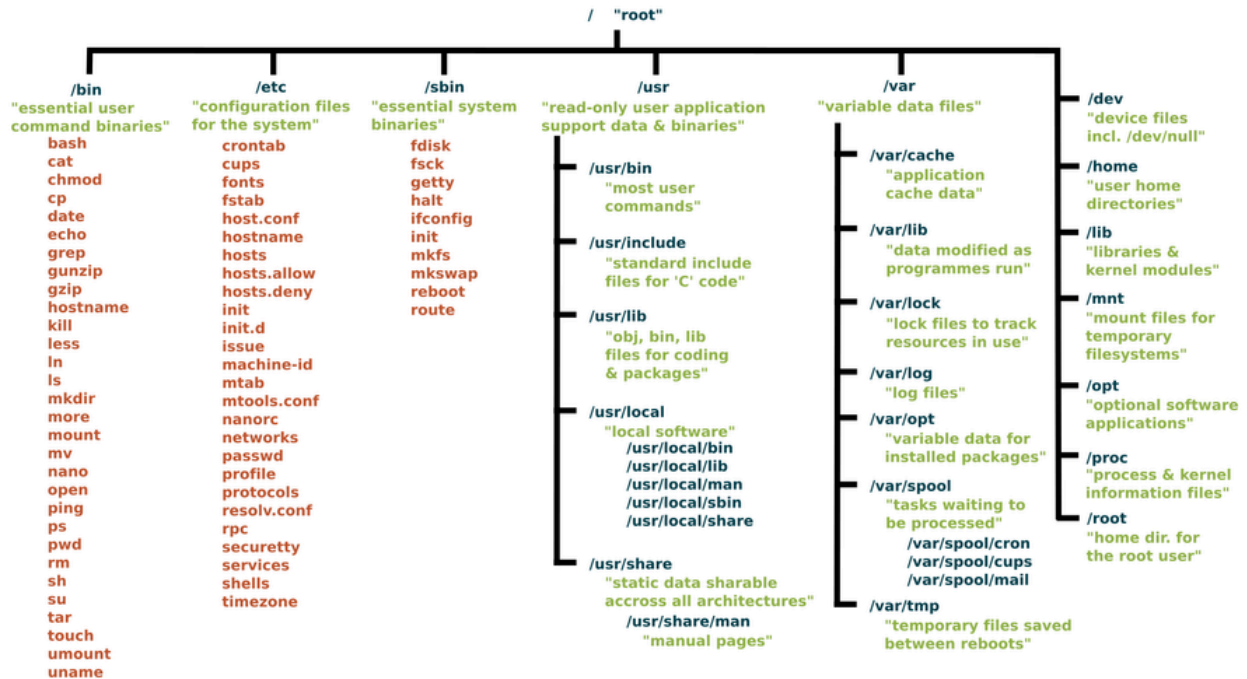
Contains temporary files created by running applications; users can also store their own temporary files here without needing superuser access.

/var

Named for its frequently changing contents, this directory holds logs in /var/log for system events and security attempts, as well as spools for tasks like print jobs and mail. It may also contain additional directories, such as /snap for software installed from snap packages.

2) Deeper look

overall idea of what the basic file system tree looks like



3) Filesystem structure

The file system handles key attributes like file names, sizes, inodes, user and group ownership, and creation dates, among others. By organizing data systematically, it enables quicker and more efficient file access.

A standard Linux distribution offers various file system options for partitioning disks, including formats like ext2, ext3, ext4, JFS, XFS, and Btrfs.

JFS

The Journaled File System (JFS) was developed by IBM for AIX UNIX as an alternative to the ext family of file systems. JFS is currently an alternative to ext4 and is known for its stability while using minimal system resources. It is particularly useful in scenarios where CPU power is limited.

XFS

XFS is a high-performance file system that extends the capabilities of JFS by supporting parallel I/O processing. Designed for high-speed data access, XFS is still used by organizations like NASA on storage servers exceeding 300 terabytes.

Btrfs

The B-Tree File System (Btrfs) is focused on features like fault tolerance, simplified management, self-repair, and support for large storage configurations. Although still under development, Btrfs is not yet recommended for production environments due to its evolving nature.

4) Create filesystem

After creating the necessary partitions, the next step is to set up filesystems on them. To view the available filesystems supported by your system, you can use ls command:

```
ls /sbin/mk*
```

The choice of filesystem should be based on your specific requirements, taking into account the benefits and limitations of each option as well as their unique features.

Two key factors to consider when selecting a filesystem are:

- Journaling support, which allows for faster data recovery in the event of a system crash.
- Security Enhanced Linux (SELinux) support, as per the project wiki, is “a security enhancement to Linux which allows users and administrators more control over access control”.

Lets create an **ext4** filesystem (supports both journaling and SELinux)

```
# mkfs -t [filesystem] -L [label] device
```

or

```
# mkfs.[filesystem] -L [label] device
```

Linux CORE: Kernel

1) The Linux Booting Process

1. BIOS

The Basic Input/Output System (BIOS) initializes hardware and loads the Master Boot Record (MBR) boot loader when the computer starts. It first checks the integrity of the HDD or SSD, then searches for the boot loader in the MBR, which may also be on a USB or CD-ROM. Once found, the BIOS loads it into memory and hands over control.

2. MBR

The Master Boot Record (MBR) is responsible for loading and executing the GRUB boot loader. It is located in the first sector of the bootable disk, usually `/dev/hda` or `/dev/sda`, and also contains information about GRUB or, in older systems, LILO.

3. GRUB

GNU GRUB (GRand Unified Bootloader) is the standard boot loader for most Linux systems. At startup, it displays a menu to select kernel options. If multiple kernels are installed, you can choose one, or the latest is selected by default. After a brief pause, GRUB loads the default kernel. Its configuration file is usually located at `/boot/grub/grub.conf` or `/etc/grub.conf`.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5PAE)
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
    initrd /boot/initrd-2.6.18-194.el5PAE.img
```

4. Kernel

The kernel is the core of any operating system, including Linux, managing all system operations. During boot, the kernel selected by GRUB mounts the root filesystem specified in `grub.conf` and runs `/sbin/init`, the first program executed with a process ID (PID) of 1. It initially uses a temporary root filesystem (`initrd`) until the actual filesystem is mounted.

5. Init

At this stage, your system begins executing runlevel programs. In traditional Linux systems, the init process would reference the init file, typically found at `/etc/inittab`, to determine the current run level.

Modern Linux distributions, however, utilize `systemd` to manage run levels. According to TecMint, the following targets correspond to the traditional run levels:

Run level 0: `poweroff.target` (with `runlevel0.target` as a symbolic link).
Run level 1: `rescue.target` (with `runlevel1.target` as a symbolic link).
Run level 3: `multi-user.target` (with `runlevel3.target` as a symbolic link).
Run level 5: `graphical.target` (with `runlevel5.target` as a symbolic link).
Run level 6: `reboot.target` (with `runlevel6.target` as a symbolic link).
Emergency mode: `emergency.target`.

After determining the appropriate target, `systemd` begins executing the corresponding runlevel programs.

6. Runlevel Programs

Depending on your specific Linux distribution, you may observe various services starting during the boot process. For example, you might see messages like "starting sendmail... OK."

These services are known as runlevel programs, and they are executed from distinct directories based on the current run level. Each of the six traditional run levels has its own directory:

Run level 0: /etc/rc0.d/

Run level 1: /etc/rc1.d/

Run level 2: /etc/rc2.d/

Run level 3: /etc/rc3.d/

Run level 4: /etc/rc4.d/

Run level 5: /etc/rc5.d/

Run level 6: /etc/rc6.d/

Keep in mind that the exact location of these directories may vary across different distributions.

Within each runlevel directory, you'll find scripts prefixed with either "S" (for startup) or "K" (for kill). The "S" scripts are executed during system startup, while the "K" scripts are invoked during shutdown.

Linux CORE: Kernel Runtime Parameters

1) Check Linux Kernel Parameters

To modify the kernel runtime parameters we will use the `sysctl` command. The exact number of parameters that can be modified can be viewed with:

```
sysctl -a | wc -l
```

If you want to view the complete list of Kernel parameters, just do:

```
sysctl -a
```

As the the output of the above command will consist of A lot of lines, we can use a pipeline followed by `less` to inspect it more carefully:

```
sysctl -a | less
```

Let's take a look at the first few lines. Please note that the first characters in each line match the names of the directories inside `/proc/sys`:

Check specific parameter using `systcl`:

```
sysctl dev.cdrom.autoclose
```

Check specific parameter printing file in `/proc`:

```
cat /proc/sys/dev/cdrom/autoclose
```

2) Modify Linux Kernel Parameters

o set the value for a kernel parameter we can also use `sysctl`, but using the `-w` option and followed by the parameter's name, the equal sign, and the desired value.

Another method consists of using `echo` to overwrite the file associated with the parameter.

```
# echo 0 > /proc/sys/net/ipv4/ip_forward  
# sysctl -w net.ipv4.ip_forward=0
```

To set these values permanently, edit `/etc/sysctl.conf` with the desired values.

```
# net.ipv4.ip_forward=0
```

Then run following command to apply the changes to the running configuration.

```
# sysctl -p
```

common troubleshooting

1) Boot Issues

Kernel Panic: If the system fails to boot with a kernel panic message, try the following:

Check hardware connections (RAM, HDD/SSD).

Boot into an older kernel from the GRUB menu.

Use a live USB to access logs and repair the filesystem.

2) Filesystem Corruption

Run fsck: Use the filesystem check utility to repair corrupted filesystems

Check dmesg Logs: Review kernel logs for disk errors:

3) Kernel Modules

List Loaded Modules: Use lsmod to see loaded kernel modules. If you suspect a module is causing issues, you can unload it:

4) Just learn more about kernel

5) Mounting Issues


Check Mount Points: Ensure that the filesystem is properly mounted. Use the mount command to see mounted filesystems:

Learn more about Kernel & Filesystems

Check TecMint - great docs!

How to Change Kernel Runtime Parameters

Introducing the /proc Filesystem

 <https://www.tecmint.com/change-modify-linux-kernel-runtime-parameters/>



Linux File System Explained

The concept of boot loading

 <https://www.tecmint.com/linux-file-system-explained/>



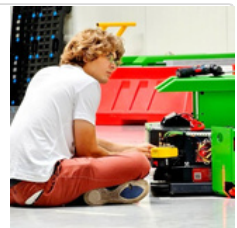
Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

<https://github.com/MichaelRobotics>



PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!