How does the Linux Kernel work?

The Linux kernel is like a bridge that allows computing communication between applications and hardware, as well as managing the system's resources.

Linus Torvalds developed the Linux kernel with C and Assembly, so he succeeded in creating a lightweight and portable core that was released to the public as open source.

You can see the Linux kernel in various fields like space, computing, smartwatches, mobile phones, robotics and health. But have you ever wondered how the Linux kernel works?

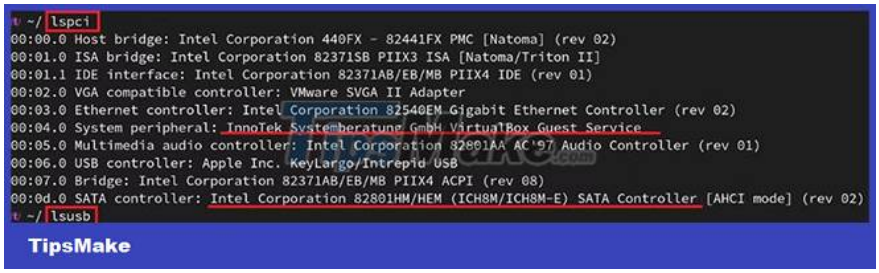
Using hardware on Linux

First of all, the Linux kernel controls which hardware to run and in what way when you turn on your computer. In addition, advanced software control is possible thanks to the programming interface. To give examples of these controls, you can view information about the hardware installed in the slots on your motherboard and benefit from this insight.

In addition, this programming interface provides an abstraction layer. For example, if you want to video chat with friends, you'll need a webcam. The abstraction layer makes it possible for the software you use to use this webcam regardless of manufacturer and model. The software here can only use interfaces that exist for Linux. The Linux kernel translates this interface's function calls into actual hardware commands that the webcam needs.

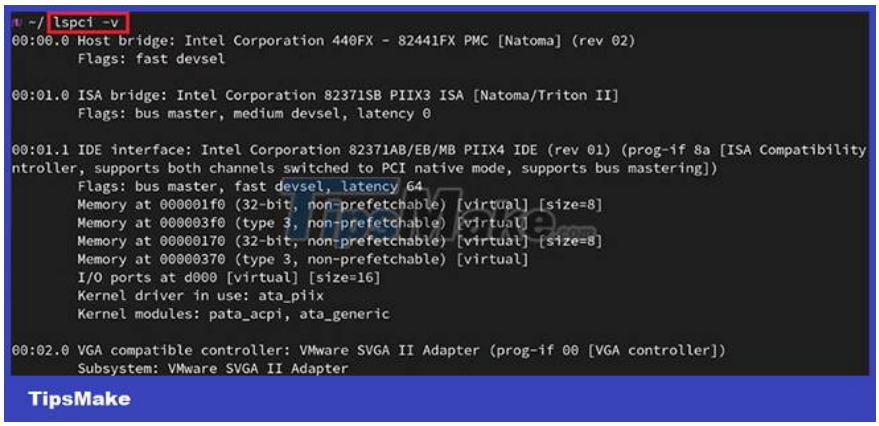
Using the /proc and /sys virtual file systems, the Linux kernel can output detailed information about the hardware it detects. Below you can see some of the tools used for this, as well as which devices and cards they output:

1. lspci: For PCI devices
2. lsusb: For USB devices
3. lspcmcia: For PCMCIA card

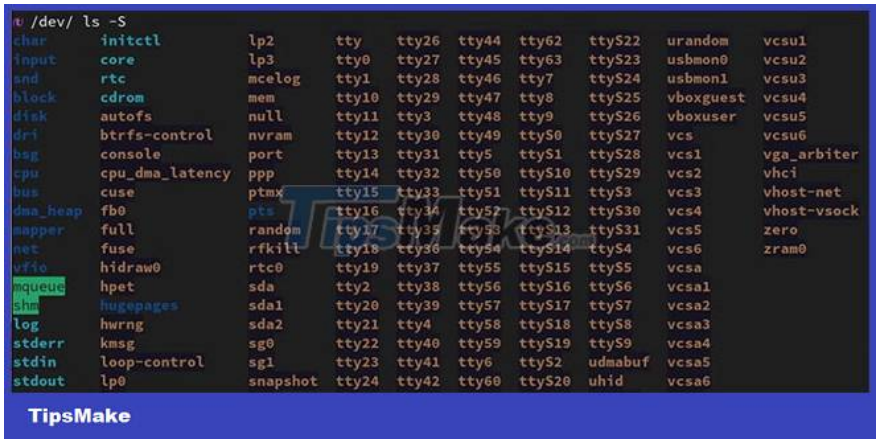


As you can see, the Linux distribution in the screenshot above runs on VirtualBox. However, you have a chance to see a lot of information like VGA controller, USB, bridge and SATA controller.

You can also use the -v parameter for more detailed information.



In the Linux kernel, applications usually access the device through special files that exist in the /dev directory. These special files represent drives and other physical devices. Files such as /dev/hda, /dev/sdc, /dev/sdc3, /dev/input/mouse0 and dev/snd/* are examples of these special files.



Linux file system management

The file system is one of the most notable components of the Linux kernel. The Linux file system is one of its biggest benefits. All files on a Linux system are gathered into a single branch. Therefore, users can use this hierarchy to get to where they want to go.

The starting point of this hierarchy is the root directory (/). The other subdirectories are under the root directory. The most used subdirectory in / is the /home directory. This subdirectory contains other subdirectories and each directory contains the actual data storage files.

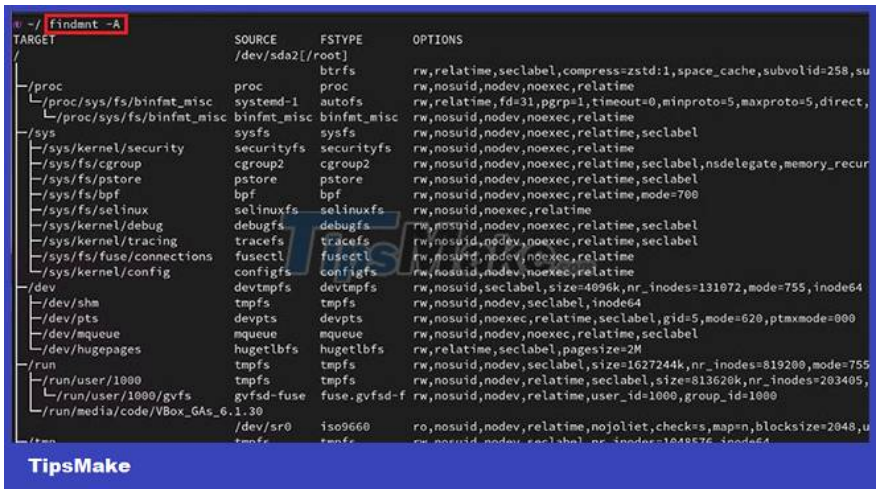
For example, you can think of a text file on the desktop. If you create a text file named helloworld.txt on your desktop, you can call it /home/muo/Desktop/helloworld.txt. The /muo example here will of course vary from case to case. As this subfolder name depends on your current username. With this naming system, the Linux kernel switches between the actual storage and the physical storage that exists on the drive.

In addition, the Linux kernel can integrate data from several drives. This is where the mount system comes into play. It uses one of the drives in the root system and mounts the others to existing directories in the hierarchy.

The other drives are then placed under the mount points. This allows users to store the /home directory on a second hard drive that also contains other subdirectories.

When you mount the drive to the /home directory, you can access these directories from normal locations. Therefore, paths like /home/muo/Desktop/helloworld.txt continue to work.

You can see the mount points between files on your system with the findmnt -A command.



With many file system formats, you can store data on the drive. The most widely known on Linux are the ext2, ext3 and ext4 file system formats. However, there are many other file system formats. In any case, you must format the file system before mounting it. You can use commands like mkfs.ext3 (mkfs stands for 'make file system' - create file system and ext3 is file system) for this.

These commands accept the device file path that you want to format as a parameter. This is a risky operation and you should use it with caution if you don't want to accidentally delete or reset the file system.

In addition, there are network file systems such as NFS that the Linux kernel uses. NFS is a network file system where data is not stored on a local drive. With NFS, data travels over the network to a server that stores the data. Since the data will reside on a server, the user does not have to constantly process it. They can also use Linux's traditional hierarchical file system as usual.

General functions

All software in a Linux system has common functions. This is why these functions are central in the Linux kernel. For example, when opening a file, you can only open it by the file name without knowing where the file is physically stored, what functions and operations it will use. All these functions are already in the kernel.

You can store your files on your hard drive, split files between multiple hard drives, or even keep files on a remote server. Shared file communication functions are important in such cases. Applications exchange data regardless of how the data is moved. The shared communication functions are there to perform these data exchange processes. This movement can be over a wireless network or even a landline phone line.

Process Management in Linux

The active version of a program that operates on data or information in memory is called a process. The task of the Linux kernel is to create and monitor these memory areas. The kernel allocates memory for a running program and loads the executable code into memory from the file system. Immediately after, the kernel runs the code.

Linux kernel supports multitasking. It is capable of running multiple processes at the same time. However, there is only one trade in any given timeframe. However, the Linux kernel divides time into small chunks, and as a result each procedure happens sequentially.

Because these small time segments are in millisecond increments, they are active only at specific times and inactive for the rest of the time. The Linux kernel's job here is to maximize performance by running multiple processes at the same time.

If the time interval is too long, the running application may not respond as you expect. If the time frame is too short, there may be problems with changing tasks. Depending on the priority of the process, the amount of time required here will vary. You may have heard of high-priority processes and low-priority processes before. This is one of the functions that the Linux kernel controls.

This explanation is not always correct. The real limitation is that there can only be one worker process per processor core at a time. Multiprocessor systems allow several processes to run in parallel. A basic system almost always has dozens of processes running.

Permissions in Linux

As with other operating systems, you can create multiple users on a Linux system. In such cases, there is a rights management system that supports individual and group users. This is where user and file permissions come into play.

The Linux kernel manages data and checks the necessary permissions for each process. For example, if you try to open a file, the kernel must check the process ID for permissions. If the kernel checks and finds that you have permissions, it will open the file.

As you can see, the Linux kernel monitors everything from file security to creating users and downloading files from the Internet. Everything is in a certain order. Every user has permissions. Linux kernel manages processes and intervals for peak performance.

Furthermore, the file system, which is one of the biggest features that distinguishes the Linux kernel from other operating systems, is very important. Linux is not something of a mystery. In contrast, all files and source code are accessible. To better understand the practical and powerful nature of the Linux kernel, you can examine the Linux directory hierarchy.