9.3.1 Hotplug and Coldplug Devices

Click one of the buttons to take you to that part of the video.

Hotplug and Coldplug Devices 0:00-0:23

In this lesson, we're going to discuss hotplug and coldplug hardware devices. Be aware that when you're working with PC hardware, and loading Linux kernel modules, you have to be aware of the differences between the PC hardware that is considered coldplug, and the PC hardware that is considered hotplug. Let's take a look what the difference is between those two.

Differences Between Hotplug and Coldplug Devices 0:24-5:35

Coldplug devices can be physically connected or disconnected from the computer system only if that system is powered off. Internal components, such as your system RAM, your system CPU, and or an expansion board that's plugged into an expansion slot, these are all the examples of internal coldplug devices. External devices can also be coldplug devices. For example, if I have a printer that connects through a parallel port, or if I have an external SCSI hard disc drive, these are examples of coldplug external devices.

The key thing to remember about coldplug devices is that fact that if you plug them in while the system is already running, they probably won't be detected and recognized by the system. On the other hand, if you do connect them properly while the system is powered off and turn the system on, they probably will be detected by the system. But, if you attempt to then unplug them while the system is running, bad things are probably going to happen. In fact, you'll probably damage the component or maybe even the motherboard itself, from an electrical standpoint.

For example, if I had a computer system running, and I opened up the case and I decided to grab a memory module and unplug it from the motherboard while the system is running, it would be bad. Likewise with the CPU, you can't just open up the case, pull out the CPU and put a different one in while the system is running. Bad things happen when you do that. Other devices are more forgiving. For example, if I were to unplug a parallel port printer from the motherboard while it was running, it probably wouldn't hurt anything, but it's still not a very good practice.

Before we go on, it is very important that you understand that there are two different types of processes that can run on a Linux system. The first type is called a kernel space process, and the second type is called a user space process.

Here's what you would need to understand, only processes that are running in the kernel space, in other words a kernel space process, is allowed to communicate directly with the system hardware. Kernel space processes are basically just part of the kernel itself.

User space processes, on the other hand, are the applications, the services, and the programs that are actually run by the end user. In order to allow user space processes to access the system hardware, we have to do things differently. They're not allowed to access the system hardware directly, instead we provide access to the system hardware through device files in the /dev directory. If a user space process needs to access a piece of hardware in your Linux system, they do so through the appropriate device file in the /dev directory. This is how user space processes access coldplug devices.

Let's say we have a user space process running, it needs to save a file to the hard disc drive, and it does so through the appropriate device file in /dev. Likewise, if a user space process needs to send a print job to a parallel port printer, it does so through the appropriate device file in /dev.

In addition to coldplug devices, there are other types of devices that may be implemented in the Linux system that are categorized as hotplug devices. Hotplug devices are designed to be connected and disconnected dynamically while the system is up and running, without any ill effects. There is software loaded on the Linux system that will detect these changes to the system as these hotplug devices are attached and detached from the motherboard. Using this software, the system will recognize when a device is connected, and when that happens it will automatically load the appropriate kernel modules to support those devices. And the same thing will happen when the device is disconnected.

Here's some examples of some common hotplug devices, we may have a USB-based printer, maybe a USB flash drive, a USB-based hard disk drive, or a camera, maybe it connects through a FireWire connection. Be aware that with a standard desktop system, the hard disk drive is not hotplug. You cannot unplug the hard disk drive and plug it back in while the system is running. However, server systems do provide hotplug storage devices, and you can add and remove hard disk drives as needed without bringing the system down. Just don't try that on a desktop system.

This brings up a problem, remember the user space processes need to access the system hardware through a device file in /dev. This isn't a problem for coldplug devices, because the appropriate device file is created when the system first boots up. But when we're dealing with hotplug devices, things are being plugged in and unplugged all the time. Can we still access the hardware from a user space process using a device file /dev? Yes, we can do it, but the process is going to need a little bit of help.

One Option to Provide Access to Hotplug Devices 5:34-6:13

There's really two approaches we can take to providing access to these hotplug devices. One option would be to just create a device file in /dev for every possible device that could ever be connected to the system. This would allow user space applications running on the system to access this hotplug hardware whenever they're connected to the system through the appropriate device file in /dev. This approach does work, and for a time many Linux distributions did this very thing. The problem with it is that it's also very, very messy because you end up with a lot of device files in /dev that frankly you don't need and will never need.

Dynamically Create a Device File in /dev 6:12-9:58

A better approach is to dynamically create a device file in /dev whenever a hotplug device is connected to the system, and then remove that device file whenever the device is disconnected. In order for this to work, the new hardware has to be recognized by the operating system itself. Then the appropriate devices files have to be created in /dev. Then any processes running on the system have to be notified that "Hey, new hardware is available, and here's the device file that you can use to access it."

In order for this to happen, several separate components are implemented in the Linux operating system. The first component is sysfs. This component provides the /sys virtual file system. And its job is to export information about the system hardware devices, both hotplug and coldplug, so that user space processes, the applications and utilities that users are running on the system are able to access the hardware in the system.

The second component that is required for this to work is the hardware abstractions layer daemon, or hald. This daemon is run automatically when the system boots up, and its job is to provide the applications that are currently running on the system with information about the hardware that's available in the system, including both hotplug and coldplug devices.

In addition to sysfs and hald, another component is required for hotplug devices to work, and that is the desktop bus, or dbus. Dbus runs as a daemon on your Linux system, and its key job, among many others that it performs, is to notify all the running processes on your system whenever a hotplug device is connected or disconnected from the system. Without this daemon, the processes running on the system would have no idea that a new hard disk drive is available on a given on a USB port.

The fourth and final component that is required to support hotplug devices on a Linux system is udev, the udev daemon, or udevd. Udev has a very important job, it's responsible for creating that virtual file system mounted at /dev that we've talked a lot about. It communicates with the Linux kernel through a special interface called uevent.

Here's the key thing that you need to remember, whenever a hotplug device is added or removed from the system the kernel sends out a uevent message, and that message is picked up by udevd. When it receives this message, it performs the task that you see here based upon the rules that are defined in the /etc/udev/rules.d directory.

The first thing it does is initialize the device, then it creates the appropriate device file automatically in the /dev directory. As we talked about, this allows the user space processes on the system to be able to access that hardware. If that new device that's been connected is a network interface, say a USB wireless network interface, then it's going to configure that interface with the ifup utility.

If, on the other hand, the new device that's been connected is a storage device, maybe a USB flash drive, then it mounts it using the information that's configured in the /etc/fstab. No matter what kind of device is connected, the last step in the process is that udev will inform all of the running processes about the new device. It basically says, "Hey guys, look we have a new hard disk drive connected," or "Hey guys, look we have a network board. Feel free to use it." By doing this the Linux operating system can support both coldplug devices as well as hotplug devices.

Summary 9:57-10:08

That's it for this lesson. In this lesson, we discussed the differences between these two different types of devices, hotplug and coldplug. We also review the mechanisms that Linux uses to enable hotplug devices to work properly with the operating system.
