6.4.2 Managing Shared Libraries

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

Manage Shared Libraries 0:00-0:07

In this demonstration, we're going to talk about managing shared libraries.

Use ldd to Identify Required Shared Libraries 0:07-1:13

In this demo, we're going to find out which shared libraries are required by a particular service running on this Linux system. Specifically, we're going to look at those required for the ntpd time synchronization daemon to run.

The first thing we need to do is find out where the executable for the ntpd daemon actually resides in the file system, so I'm going to switch to my root user account and I'm going to run the 'which ntdp' command. We see that the ntpd executable, which is run to execute the ntpd service, is located in the /sbin directory.

Now that we know where the executable is, we need to determine which shared libraries are required by this executable in order for it to run, because this executable does not have all the software--all the coding--necessary for it to run. It is relying upon code that's going to be borrowed from these shared libraries. If those shared libraries aren't there, it can't run.

To find out which shared libraries are actually required by this executable, we enter 'ldd /sbin/ntpd'.

Identify and Install Missing Shared Libraries 1:13-3:26

Here it gives us a list of all of the different shared libraries, along with their location over here, that are required in order for this executable to run and for the ntp daemon to be enabled on the system.

What we're going to do at this point is deliberately corrupt one of these shared files. We won't actually damage it. All we're going to do is just rename it so that it's not available. This is going to cause problems for the ntp daemon, because one of the shared libraries that it relies upon is about to become unavailable.

I'm going to use the move command to rename this file right here. I'll enter 'mv /usr/', and we look over and we can see that this file resides in the lib64 subdirectory, 'lib64/libcrypto.so.10'. We're going to rename it '/usr/lib64/libcrypto.so.10.old'.

Now that we've made that file name change, let's run the 'ldd' command again for the ntp daemon, and now notice that we've got a problem. It says that the ntp daemon requires the libcrypto.so.10 file, but it's not found, so we're going to have problems if we try to run the ntp daemon on this system.

As you can see, the ldd command can be a really useful tool for troubleshooting issues with shared libraries. If you've got an application or a daemon that won't run and it reports that shared library files are missing, you can use the ldd command to find out exactly which ones are missing and then locate the appropriate software packages that contains those shared libraries and install them and all will be well once again in the world.

Let's simulate doing just that. Say we found out that libcrypto.so.10 is required, and we went and did some research, found out which software package contains that file. We've installed it on the system now, but really what we're going to do is just rename it back to its old name.

We'll rename the .old file back to its original file name. We'll run the 'ldd' command again and everything is well. All of the shared library files are available that ntpd needs in order to run.

Summary 3:27-3:39

That's it for this demonstration. In this demo, we talked about managing shared libraries. We talked about how to use ldd to identify which shared libraries are required for a particular executable to run, and then we talked about the process for identifying missing shared libraries and installing them.
