8.12.6 The cpio and dd Commands

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

cpio and dd 0:00-0:44

In this lesson, we're going to look at two very handy utilities that you can use to manage the files and directories in your Linux file system. The first one is the cpio command, or cpio, and the other command is the dd command.

Let's begin by looking at cpio. The cpio utility is a backup and archiving utility. You can use it to make an archive file just like you can with the tar utility; however, there are some key differences between the two commands.

For example, with the cpio command, you have to send cpio a list of files that you want added to the archive, to the cpio command's standard input. And that's not the case with the tar command.

Create a cpio Archive 0:45-4:51

With tar, you can specify as a command option what directory or file you want to add to the archive. With cpio, we have to do things quite differently.

One way to send cpio a list of files to backup is to use the cat command to display the contents of a text file that contains a list of files to be included in the archive. We'd run cat, followed by the file name, and we would pipe the output of cat to the input of the cpio command.

Another way to do this is to use the find command or the ls command to generate a list of files that you want to add to the archive.

Using cat to display the contents of a text file is useful in situations where you have lots of files spread all the way through the file system that you want included in the archive.

Using find is useful in situations where you want to look through the file system for specific files that match a particular pattern, say all files that have an extension of .odt.

The ls command is useful in situations where you want to add files to a cpio archive that exist all in the same directory structure.

For example, let's suppose I want to back up the contents of my user's /home directory. I'm the rtracy user. So in order to do this with the cpio command, I first have to somehow generate a listing of all the files and all the directories within the '/home/rtracy' directory and send that list to the input--the standard in--of the cpio command.

Once we do that, we then have to redirect the output from the cpio command to a file in the file system.

One way to do this is to use the find command. First, I switch to the rtracy user's home directory /home/rtracy, and then I run find. I specify a period here to say start in the current directory. And then I specify -depth and then I specify -print. When I do this, the find utility is going to generate a listing of all the files in the current directory.

The important point here is that because I used the -print option, find will simply print the full name of each file to its standard output. Because I used the -depth option, the find command is going to go through the contents of the current directory first and then it'll go through each subdirectory of /home/rtracy and print those file names as well.

You might be asking, "Why did you use find and not the ls command? Wouldn't the ls command work in this situation?"

Actually, you could. We could have used ls to look recursively at all the subdirectories in the /home/rtracy directory and generate a listing of files that way as well. Either option would work.

Notice here that we've taken the output of the find command, and we've piped it to the input of the cpio command. The cpio utility will then use this list of files and directories to create the archive.

We specify -o option with cpio to tell it to create a new archive and we use the v option just to do so verbosely, so it displays the name of each file and directory being added to the archive out of this process. You don't have to use the v option. I like using it because I like to see what's going on.

Here's a quirky thing you need to understand about the cpio command, and that is the fact that you have to redirect the output to a file in the file system. If you do not, if you omit the greater than sign, all cpio is going to do is print a list of files that it wants to add to the archive to the screen. It doesn't actually create the archive file.

So, what we have to do is redirect the standard out from the cpio command to a specific file in the file system. In this case, I probably have a USB drive mounted in the /media/usb directory, and I'm going to create a new backup file, a cpio archive file, on that USB drive named backup.cpio.

Compress a cpio Archive 4:49-6:21

When you create an archive file with tar, you have the option of compressing that archive file to shrink its size down. That's a really good thing, especially if you're backing up users' home directories, because I can guarantee you they've got music and video files in there that are big and are going to consume a lot of space.

You can do the same thing with a cpio archive, it's a little bit different though than the way you do it with tar. You compress a cpio archive by adding the gzip utility to the pipe. We've done that here. We've used the same find command that we used before to create a listing of all the files in my user's home directory, /home/rtracy.

Then we're going to pipe the output to the input--the standard in of the cpio command--and we're going to use the -o and -v options, just like we did before, to verbosely create a new archive file.

Instead of redirecting the output of cpio to the backup file like we did before, we're instead going to pipe the output of cpio to the input of the gzip command, which will then compress the archive file. And we redirect the output of the gzip command to the actual file in the file system.

Notice over here that I added an extension onto the end of the archive file name '.gz'. That way, when I go to extract files from this archive I will look at it and say, "Oh, I compressed that with the gzip utility. Therefore, before I can extract files from it, I first have to decompress it with the gunzip utility. Then I can use cpio to pull the files out of it."

Extract Files from a cpio Archive 6:21-7:15

Let's suppose I now need to restore files from my cpio archive. I need to take that archive file and pull the files back out of it. If I haven't compressed it, I would use the command shown here. I'd run cpio and then I use the -i option followed by the -v option.

This tells cpio to verbosely extract files from the file name that's being sent to the standard input of the command. Notice here I've used the less than sign (<) to send this file name to the standard in of the cpio command.

That's why cpio can be a little bit confusing. Everything you want to do with cpio, whether creating an archive or extracting from an archive, has to go to its standard input.

When I run the command, then the files and directories within the archive are extracted to the current directory, which, as you can see here, is a /tmp directory. All the files from my rtracy user's home directory will be extracted to /tmp.

Extract Files from a Compressed cpio Archive 7:14-8:44

As I said just a minute ago, if the cpio archive was compressed with the gzip utility, we have to decompress it first before we can use cpio to access the files within it. In this case, I've used the 'gunzip' command to decompress the backup.cpio.gz archive file.

The resulting file is the actual cpio archive backup.cpio--and we can now use the cpio command to extract files from this archive. Be aware that there is actually another way to do this.

What you can do is add the -c option that you see here to the gunzip command, and then we specify the name of the compressed archive file that we want to decompress. And then we pipe the output of the gunzip command to the input of the cpio command with the -i option telling it to extract files.

Basically, this -c option tells gunzip to write the output of the command to the standard output, which basically leaves the original file intact.

Essentially, this just writes the uncompressed file name to the standard out, which is then set to cpio for processing. Essentially, the gunzip command then extracts the name of the file compressed in the gzip archive and sends it to the cpio -i command, which then extracts the file out of the archive to the local directory and then cpio processes it.

Copy a File with dd 8:44-10:39

Before we end this lesson, I want to spend some time looking at another very, very useful file system utility that's available on Linux. It's called the dd utility. I actually use dd all the time.

It's great for doing a variety of different tasks, from disk imaging, to forensics, to backing up master boot records. There's many different things you can do with dd.

Basically, the dd command is used to copy files and you can copy all kinds of Linux data with this command. For example, you can use dd to copy an entire disk partition. You're probably thinking at this point, "Big deal. I already know how to use the cp command and the mv command. Why do I need to know about the dd command?"

Actually, dd is really quite useful. The key difference between dd and the other file copy utilities, is the fact that dd copies data using records. The default file size for a record with dd is 512 bytes, so if I wanted to use dd to just copy a file in the file system, the syntax is pretty easy.

I enter dd and then I enter if= and the name of the input file, space, then of= and the name of the output file. I'm going to copy this file to this file.

An example of doing this step is shown down here. I entered dd and then for the input file I'm going to specify the boot.log file that resides in my home directory. Then for the output file I'm going to write it to a file named boot.log.copy in the /tmp directory. Then the copy process proceeds as you can see here.

Do I use dd to do this? Actually, no, not very often because I'm not doing anything special when I'm just copying a file with dd. I could do the same thing using cp. Or, if I wanted to move it, I could use the mv command. You might be asking, "Well, when would you use dd if you don't use it to copy files?"

Copy a Partition with dd 10:39-12:23

I use dd all the time to do non-traditional file copying. For example, because dd uses records to copy data, I can use it to copy an entire disk partition to a single file, and this is really powerful. Essentially what dd allows me to do, is use a simple command line tool to implement a very powerful form of drive imaging.

What I can do is use dd to create an image file of a disk partition and then I can take that file, move it to another system, and then use dd to extract that image file to create a new disk partition on a blank hard disk drive. In essence, by doing that, I cloned or copied the entire partition from one disk to the other.

The reason this works is because dd will copy all of the records, or blocks, within that partition--whether there's any data associated with those blocks or not. Basically, it creates an exact copy of that partition in that image file. When you restore it to a different hard disk drive, you got an exact copy--bit-by-bit--of the original partition.

The syntax for doing this is to enter dd and then, for your if parameter, specify the device file name of the partition that you want to copy, and then the name of the output file you want to create.

In this example, I use the dd command to copy the entire /dev/sdb1 partition to a file named partbak.dd on a USB drive, and it does take a little bit of time to complete because the partition's going to be very large. But then, I can take that file and use the dd command to restore that partition onto another hard disk drive.

Copy a Disk with dd 12:21-12:57

I can even use the dd command to create an image file of an entire hard disk drive. The syntax is pretty much the same. Instead of specifying the device file of a partition, I use the device file for the hard disk itself.

In this command, I'm taking the entire sdb hard disk drive and copying it bit-for-bit to a file named driveback.dd on an external USB drive, and then I can use that to clone that hard disk drive onto another disk using the dd command. I would just reverse the input and the output file names when I restore.

Back Up the MBR with dd 12:55-14:20

The last thing I'm going to show you how to do with dd is to backup the master boot record. If your hard drives use the MBR partitioning scheme, then you can create a backup copy of that drive's master boot record and its partition table. This is possible, again, because the dd command looks at data as records.

The syntax is shown here. I enter dd and then the input file is the device's file name. Then the output file is the name of the file we want to create, nothing new there; but, this is where things are different.

First, I use the bs parameter to specify how big I want the record size to be. On a disk that uses MBR partitioning, the first 512K block of the hard drive is where the master boot record and the partition table reside, so I'm going to tell dd to grab the first 512K block of this device file.

And I use a count=1 parameter to just get the first one, just grab the first 512K block, and I did that down here.

I'm looking at the main hard drive in the system /dev/sda. Here's the file I'm going to back up the master boot record to. I specify a record size of 512K, and I say I just want the first 512K block. That backs up my master boot record to this file.

Summary 14:20-14:28

That's it for this lesson. In this lesson, we talked about how you can use the cpio command to create and manage archive files and we also reviewed how you can use the dd command to copy file system data.
