8.12.7 Using cpio and dd

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

Using cpio and dd 0:00-0:10

In this demonstration we're going to practice using the cpio and dd utilities. Let's begin by using the cpio command.

cpio 0:11-4:21

The first quirky thing about cpio that you need to be familiar with is the fact that you need to send a stream of file names that you want to archive to the standard in of the command. Then you need to redirect the output of the command to a file in the file system. Let's see how this works. I'll switch to my root user account. Let's switch to the /mnt/shared/Rand directory. I'll use the 'ls' command to generate a listing of all the files that are currently in this directory.

Let's suppose we need to archive these files; this is a shared directory used by many users and we're worried that somebody is going to accidentally delete a file and need it back, so we're going to create an archive file every so often, using cpio that contains the current version of these files so that if the user does accidentally delete one, we can go and grab it out of the archive, extract it, and put it back in the directory where it originally resided.

As I said a minute ago, we need to send a list of these file names to the input of the cpio command in order for it to create its archive. One way to do that is to use the 'ls' command just like we did a minute ago. We need to pipe the output of the ls command to the input of the 'cpio' command. By doing this, ls will generate this list of files that you see here, send it to the input of cpio; cpio will grab each one of those files, and then add it to the archive.

We need to specify the '-o' option which creates a new archive. I'm also going to add the '-v' option, which causes cpio to run in verbose mode, then this will cause cpio to display each file that it adds to the archive, just so we can verify that everything worked properly.

As I said earlier, the second quirk with 'cpio' is if you run the command as you see it now, all it will do is write the output of the 'cpio' command to the screen--not terribly useful when we're trying to create an archive. Instead, we need redirect the output of cpio to a file in the file system. I'm going to put the output file in my root user's home directory, so I'll specify, '~', and let's give it a name of 'RandD_bak'. Then I'm going to add an extension of the file of 'cpio'. This tells me that this archive file was created with the cpio utility. That way when I look at it six months down the road and I can't remember why and how I created that file, I'll look at it and say, "Oh, yeah, that's a cpio archive, so I need to use cpio in order to extract any files from it".

Press 'Enter', we see a list of files as they were added to the archive, now, I'll use the 'ls' command to view the contents of my home directory and we should see the RandD_bak.cpio file. Okay, it worked. Let's go ahead and switch to my home directory now. Let's suppose now that a user did accidentally delete all the files in this shared RandD directory, we've got to get them back as soon as possible, so I'm going to extract these files from the archive that I just created. I'm just going to extract them to my home directory for now and then I would copy them over to the destination directory later on.

To do this we need to run 'cpio' again. This time we need to specify '-i'. '-i' tells it to extract, whereas '-o' causes it to add files to the archive. I'm going to cause it to run in verbose mode again so I can see what's going on. There's a third quirky thing that you need to be familiar with, with cpio. If you want to extract files from an existing archive, you need to send the name of the archive file to the standard input of cpio, just like we did before when we were adding files to an archive.

One way to do this is to use the 'ls' command just like we did before. You can run ls -RandD_bak.cpio and pipe the output to the input of the cpio command. That would work. Another way is to redirect the input of the command. We use the less than '<' sign, and then the name of the file that we want to send to the standard in of the 'cpio' command, 'RandD_bak.cpio', 'Enter'. If I run the 'ls' command, I see all the files that were pulled out of the archive file. Then I can copy those all back over to the RandD folder in the shared directory. To rectify the problem that my end users caused. That is how you use cpio.

Advantages of dd Utility 4:22-5:22

Let's shift gears and look at the dd utility. As I said in the beginning of this demonstration, the dd utility is used to copy data. You might be asking, "Why would I want to use dd if I already have cp and mv for copy and move?" dd does copy data, but it does it very differently, and more powerfully I might add, than the cp and mv command, because cp and mv only work on files. dd does not care really that much about files; instead, it copies on a record basis. It can copy individual records from the hard disc drive, each record being 512 bytes each. By doing this, it can copy data that you cannot copy with the 'cp' command. For example, you could copy an entire hard disc. Not just the files on the hard disc, but the actual entire hard disc, including all the unallocated areas on the disc. You can do it to copy an entire partition. You can use it to copy your boot records, and so on.

Syntax of dd 5:23-6:30

Now the syntax for using dd is to type 'dd' at the command prompt. Then you use 'if=' for input file to specify what it is you want to copy. Then you use the 'of=' option to specify where you want to copy it to. There are a couple of other useful options you can use as well with the dd command. For example, you could enter 'bs=' to specify how many bytes to copy at a time. You can also use the 'count=' parameter to specify how many blocks to copy before dd quits.

In this demonstration we are not going to copy regular files with dd, because frankly it's a lot easier to copy files with cp. Where dd really shines is when you need to copy data that you cannot copy with cp. For example, I'm going to erase my command here. I'm going to run the 'mount' command first and show you that, currently we have the /dev/sdb1' partition mounted in the '/mnt/shared' directory. If we do an 'ls' command of '/mnt/shared', we can see all the files and subdirectories that reside on the disk partition.

Copy Entire Partitions to a Single File 6:31-8:14

We can use the dd command to copy the entire sdb1 partition to a single file. We can put it anywhere we want. We can even store it in my root user's home directory. To do this, I would type 'dd', and then we have to specify the input file, 'if=', and we're going to specify '/dev/sdb1'. By specifying the device file for the partition as the input file, we're going to use dd to copy this entire partition. Every single block within this partition, including those that are blank that have no content in them. The entire partition will be copied. Then we have to specify where to copy it. The output file will be '/root/sdb1_img'. Essentially, what we're doing is making a copy of this entire partition. We're creating an image, really, of this partition in a single file. With this image file, you could then use dd again to copy that image file out to a partition on a different hard disk. Essentially, you can use dd as a total imaging solution for managing hard disc imaging in your organization. It works really well for that. I'm going to press 'Enter', then I'll pause the recording because this will take a little while to complete.

All right, the imaging process that we started with dd is now complete. You can see it took the entire partition, about 11GBs, and copied it to an image file. If we do an 'ls' command here, here's my image file; I suspect that if we do an 'ls -l' we'll see that it's a very large file that is between 10 and 11 GB in size.

Backing Up MBR 8:15-9:49

You can also use the dd command to back up your master boot record if you're using MBR partitions on your Linux system. This can be a useful thing, because if your MBR gets messed up, then you lose access to your partitions. One way to back up your MBR is to use the 'dd' command again, and we specify input file this time of the hard disc itself, where my Linux operating system is installed, in this case I'm installed on '/dev/sda'.

I don't specify a partition, just the hard disc itself, because that's where the master boot record resides. For the output file we'll save it in my root user's home directory again and we'll specify a name of 'mbr.bak'. If I hit 'Enter' at this point, what will happen is the entire hard disc drive--an image of the entire hard disc drive--will be written to this file, and that would be a very large file and that's not what I want to do. All I want is the very first block of the hard disc drive itself, because that's where the MBR is stored. So I use the 'bs=' option, which specifies the number of bytes to copy. I'm going to specify '512' to copy the first block. Then I use the 'count' option to specify that I want only the first block of the hard disc drive to be copied. Press 'Enter'. If I do an 'ls' command, in my home directory, we see that I've created a backup of the very first block of my hard disc drive containing my master boot record. dd just grabbed the first 512 block from the beginning of the SD hard disc drive and stored it in this file.

Summary 9:50-10:06

That's it for this demonstration. In this demo we talked about using the cpio and dd archives. We first used cpio to create an archive backup of a shared folder, then we used the dd utility to make an image of the disc partition, and then to also back up the master boot record of a hard disc drive.
