8.1.2 Managing MBR Partitions

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

Manage MBR Partitions 0:00-0:27

In this demonstration we're going to talk about managing MBR disk partitions. We're going to do it today using the fdisk utility.

However, the rtracy user is a standard user, and I cannot use the fdisk utility. I have to elevate my privileges to root first. We'll use the 'su' command to do that. Now we'll run the 'fdisk' command.

Create a Primary Partition Using the fdisk Utility 0:28-5:47

The syntax is fdisk and then a space and then the name of the device file representing the hard disk drive that you want to manage partitions on.

In this scenario it is the second hard disk drive in the system, so the device file is /dev/sdb. sda is the hard disk where my Linux operating system is installed and my home directories reside, and so on. We don't want to touch that disk. We want to work with the new hard disk drive in the system.

We press Enter. A couple of warnings are displayed when we do. First of all, it tells us the changes that we make with the fdisk command will be stored in memory only until we actually write them to disk.

We can make all the changes we want with fdisk, but until we actually write them, no changes are actually made to the disk.

I do need to point out at this point that if this disk already had partitions on it and we're going to start removing partitions, re-creating partitions and so on, you should definitely backup any desirable data off of that disk first because repartitioning a disk will destroy that data and make it unavailable.

Also we have a warning here telling us that the device does not contain a recognized partition table. That's actually not a big problem. It's a brand new disk and there are no partitions on it and therefore it has no partition table and that's okay. It tells us that because it didn't have a partition table it created one of its own. We're happy with that. We needed it to do that.

The first thing I want to do is use the 'm' command as indicated right here to view a list of the various commands that I can use with the fdisk utility. To begin with, let's use the 'p' command right here to just view a listing of all the partitions that may already exist on the drive. I know there aren't any, but if there were, it would be a good idea to know what we're working with first.

The output of the 'p' command is here. It tells us the name of the disk. How big it is and so on. The number of sectors and notice there are no partitions listed. We know that it's a blank disk. We don't have to remove any partitions in order to create new ones.

We're good to go and if I scroll up here, we can see that the n command is used to add a new disk partition. Let's go ahead and do that.

Go down here. Enter 'n'. Now we have to decide what type of partition we want to create. Do we want to create a primary partition or do we want to create an extended partition which, as it notes here, is a container for further logical partitions.

For our first partition on the disk let's just go ahead and create a primary partition. We will enter 'p'. It asks us to specify which partition number this will be. It's the first one so we'll use '1'.

We have to specify where on the hard disk drive this partition is going to reside. To do this we have to specify, first of all, the first sector that will be included in the partition. Let's go ahead and use the default of 2048, that's actually the first available sector on the disk so we're going to tuck to partition right at the beginning of the disk. Press Enter.

Now we have to specify what the last sector will be and there are lots of different options we can choose to do this. If you really like math you can calculate the number of sectors that you want to use.

To do this, you have to understand that there are 512 bytes in a sector, so using that figure and the overall size that you want to create the disk, you could mathematically determine what the ending sector would be. Or, if you're lazy like me, you could just manually specify how big you want it to be.

To do that we enter a (plus) '+' sign and then we specify how big we want the partition to be, and we can specify the size in terms of kilobytes, megabytes, gigabytes, terabytes and so on. This is just a little 20GB hard disk drive that we've connected to the system and I want to use half of it for the primary partition, so let's enter '10 G' for 10 GB.

If we do the 'p' command to print the partition table you can see that the first partition is now /dev/sdb1. Here's the start sector. Here's the end sector. Here's the total number of sectors, and here's the size. Note over here that there's an ID and type parameter associated with this partition.

Now you may from time to time need to actually change the partition type. For example, we may have just created a partition on this disk and we actually don't want to use it to store data but instead want to use it for a swap partition.

Alternatively, say if this weren't an external hard disk drive we may want the partitions on this drive to be readable by other operating system so we might want to change the partition type such that it can be read on, say, a Windows system.

To do this, you use the 't' command here to change the partition type. Now you have to know what type you want to change it to because you'll have to specify the numeric code of the partition type you want to use. If you don't know what that is, you can use the 'l' command right here to list known partition types.

Let's do that real quick. Here you can see that if we wanted to switch it to a swap partition we would specify a code of 82. On the other hand, if we wanted to create an NTFS partition that would be readable on a Windows system, we would use this ID right here, 70, to create an NTFS partition. Or we could use 'b' to create a FAT32 partition, which would have even broader compatibility.

For our purposes here, however, we're just going to be using this on a Linux system and we're going to use that to store Linux data, therefore type '83' is what we want to use. Okay, so we've created one partition on the disk.

Create an Extended Partition Using the fdisk Utility 5:48-7:16

Now we need to create a second partition. We'll enter 'n' to create a new partition again. This time we are going to create a different type of partition. Instead of creating a primary, we're going to create an extended partition. We'll set this to partition number '2' and then we have to specify where that partition is going to reside on the disk.

We'll begin at the next available sector. You can see that the ending sector for sdb1 is 20973567, so we'll start the next partition on the next available sector, 20973568.

Use the default and then we just want to use the rest of the disk so rather than specifying a size, I'll just go ahead and use the default last sector available on the hard drive. Press Enter, and it tells us that we created another 10 GB partition that is an extended partition. Now if I hit 'p', you can see that we have a standard regular partition here but now we have an extended partition.

Now extended partitions are really useful if you're working with MBR partitions. The reason for that is because with the MBR partitioning scheme, you can have only a maximum of four partitions on any given disk. There are many times when you are going to need a lot more partitions then just four.

What we do is create one single extended partition and then we can find many, many, many logical partitions inside of it. That gets us around that four partition limit.

Create Logical Partitions Within Extended Partitions 7:17-10:11

Let's create a couple of logical partitions. I'll enter 'n' again. Notice here that I was not prompted to specify what type of partition I wanted to create because I've already used up all the available space with my primary and extended partitions. The fdisk utility knows that the only type of partition I can create now is a logical partition.

It's going to add a logical partition and its number is going to be five. Any partition number that you see that is greater than four, you automatically know is a logical partition, as long as you are using the MBR partitioning scheme.

Now we have to specify how big we want that logical partition to be. We'll start at the first available sector and then let's just do '+5G' to make it 5 GB in size.

Now if we do a 'p' we can see the new logical partition that's been created. Let's go ahead and create one more. I'll do an 'n' to create a new partition. We'll start at the first available sector within the extended partition and then we'll just use the rest of the space within the extended partition rather than specifying a size. Now if we do a 'p', now we see that we have two logical partitions created within our extended partition.

As we said at the very beginning of this demonstration, the fact that we've defined these partitions means nothing in terms of the hard disk itself because we haven't actually committed them to the hard disk drive. We are basically designing how we want our partitions to work and only when we're satisfied do we actually commit them to disk.

At this point I'm very happy with this partitioning scheme so let's go ahead and press 'w' to write out the changes to disk.

At this point, the changes have been written to disk. These partitions now exist on the hard disk drive.

Depending upon the distribution, the kernel may or may not be aware of the new partitions that you've created on that hard disk drive. The fdisk utility on most newer distributions will notify the kernel when new partitions have been created so the kernel knows they're there and you can now use other utilities to create file systems on them and melt them in the file system.

However, the fdisk utility on some older distributions does not notify the kernel that the disk has been partitioned and there are new partitions available. One option for making the kernel aware is just to reboot the system because when the system boots, the kernel will see the new partitions and make them available.

However, if you don't want to do that, you can simply use the 'partprobe' command, which stands for partition probe. It will go out and check the disks for any new partitions that the kernel doesn't know about and make it aware. We will run it here, even though I don't think we actually needed on this distribution.

At this point, we've created three data partitions that we can write data to. However, we can't do that yet. Before you can actually write data to these partitions, we have to format them with the file system and then we have to mount them somewhere within the file system hierarchy on the system so we can actually access them. We are not going to do that here. We'll do that in a separate demonstration.

Summary 10:12-10:27

That's it for this demonstration. In this demo we talked about how to manage MBR partitioning. We first used the fdisk utility to create a primary partition. We then used the fdisk utility to create an extended partition, and then we ended this demonstration by using the fdisk utility to create logical partitions within an extended partition.
