8.2.1 GUID Partitions

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

GUID Partitions 0:00-0:17

In this lesson, we are going to discuss how to manage GUID partitions on a Linux system. The GUID partitioning scheme is a newer partitioning scheme, and we need to discuss why you would want to use GUID partitions over older partitioning schemes, such as MBR.

MBR Limitations 0:18-1:42

The key problem here is that MBR has a lot of limitations. It's been around for a very, very long time. It's been around since the early 1980s. And while it was great in its day, today it's got a lot of limitations and a lot of short comings. Here are some examples.

First of all, the master boot record under MBR has to be installed in the first 512 bytes of the hard disk drive. You can have only four primary partitions on a single storage device, and that is really limiting.

Also, the default block size used by MBR is 512 bytes. Because of this, the MBR partition on the hard drive can be up to only two terabytes in size. In the 1980s, that was a huge amount of space--almost an unthinkable amount of space. Terabyte drives are cheap. You can go down to your local big box store and buy one.

To address these limitations, many patches, workarounds, and really downright kludges have been implemented over the years and tried to get around these MBR shortcomings, such as using logical block addressing or LBA. This allows you to use larger hard disk drives.

In addition, we have started using 4 KB sectors with MBR partitions in order to increase the maximum partition size on a particular disk. We have also started defining one of our available four partitions as an extended partition so it can hold many logical partitions.

---

gdisk Utility 1:43-7:52

All of these MBR work-arounds have served us well for a really long time but, frankly, they have outlived their usefulness. We need bigger disks, and we need better performance out of our disks.

Accordingly, a new disk partitioning scheme is now available in Linux. It's called the Globally Unique Identifier Partition Table Scheme. We just call it GUID, or sometimes just GPT. It's intended to be a replacement for the old MBR partitioning scheme.

By the way, GPT stands for GUID Partition Table. Really, GPT is a part of the overall Unified Extensible Firmware Interface, or UEFI specification, which is designed, really, to replace the BIOS on most computer systems and provide a better software to hardware interface. Basically, think of UEFI as a BIOS on steroids.

Even though UEFI is implemented only on very late model computers, you can still use GPT partitioning on older Linux systems that still use a standard BIOS. Really, GPT is a better way to go versus MBR and that's because it has several key advantages that you see here.

First of all, there are no longer such things as a primary, extended, or logical partition if we're dealing with GUID partitioning. All the partitions in GUID are just plain old partitions. They are all the same.

In addition, GPT supports gigantic disk partitions. We're talking about partitions that are measured in zebibytes instead of terabytes. Remember, with MBR we were limited to four partitions per disk. Well, with GPT, you can have 128 partitions per disk. And that's a lot more than four.

In addition, GPT provides fault tolerance by storing a copy of the partition table in the first and last sectors on the disk. This is a really good idea because if one of the copies gets corrupted, and this has happened to me before, then the redundant copy can take over and the files on the disk remain available. I had this happen on an MBR-based system. Once the master boot record is corrupt, you are toast.

In addition, GPT performs a cyclic redundancy check, or CRC, to verify the integrity of the partition table. It also signs unique IDs to every hard disk and every partition on each hard disk.

To manage GPT partitions, we can't use fdisk anymore. Instead, we use a utility named gdisk, which works in many ways like fdisk. You can use gdisk to do a lot of different things. You can use it to convert an MBR partition table to a GPT partition table. You can use gdisk to verify a hard disk drive.

Of course, you can use gdisk to create and delete GPT partitions. You can use gdisk to display information about a partition. You can use gdisk to change the name of the type of a partition. You can also use gdisk to backup and restore a hard disk partition table.

For example, suppose I added a second hard disk drive to my system, and I wanted to create a GPT partition on it so I can use it to store data. In order to do this, I would enter gdisk at the shell prompt and then the device file I wanted to manage partitions on /dev/sdb.

The syntax is the same as the fdisk utility. Really, if you are comfortable using fdisk, then using gdisk will be really easy because, really, the commands are almost the same.

It's a good idea when you first start using gdisk to use the question mark (?) command to get a list of all the available commands. For example, you can use d to delete a partition; l to show a list of known partitions types; n of course creates a new partition; and p prints the partition table; q allows you to quit without saving changes, which is a really useful thing.

If you're going and creating partitions, and realize you really don't like what's happened, you hit q and everything goes back to the way it was because, just like fdisk, gdisk utility does not actually write the changes to disk immediately.

The changes are only saved in memory until you actually commit them with the w command. You can use the t command as well to change partitions type.

If you wanted to create a new partition on the disk, you would enter n at the gdisk prompt, and then you specify the partition number. You specify your first sector that you want to use for the partition. Then you specify the last sector you want to use for the partition.

By default, what gdisk is going to do is pick the next available sector for the first sector. In this situation, there are no partitions already on the disk, so I picked the very first sector available.

Then, for the last sector, you can either specify the sector you want to use, which I don't know of anybody who actually does. What we usually do is just specify the size that we want to use. In this case, I specified 10 gigs.

Notice, also, you are given the option to specify the partition type you want to use for the partition. The partition type numbers that you use with gdisk are very, very similar to those used with MBR partitions. Notice here that in order to create a standard Linux partition, we use the type code of 8300. That's the default. Using MBR partitions, the type code for a standard Linux partition is 83. You can see there is a lot of parallels there.

Just as with fdisk for MBR partitions with gdisk, you can use the l command at the gdisk prompt to view a list of all possible partition types, and their associated codes.

Once you have all of your partitions defined, you can use the p command to view a list of all the partitions on the disk. If you decide you want to delete one of these partitions for some reason, you just use the 'd' command--just like with fdisk.

You need to change its type, use the 't' command. If you decide you want to back out and to start over enter 'q' to abandon all your changes. If you do like your changes, and you want to commit them to disk, use the w command.

---

parted Utility 7:53-10:41

In addition to the gdisk command, you can also use the parted command at the shell prompt to manage GPT partitions. parted stands for partition editor. You can use parted to add partitions, delete partitions, and edit partitions on your disk.

Before we look at how you use parted, there is a very important thing that you must understand about how parted works. Unlike gdisk and fdisk for that matter, the parted command will write the partition changes you specify immediately to the disk.

You don't have the option of backing out like you do with fdisk and gdisk. You need to be absolutely certain of the changes you want to make before you start using parted.

To use parted, you just enter the parted command at the shell prompt. You use the select command to specify which hard disk you want to actually work on. You have to be kind of careful with this because, if you don't specify a disk to manage, notice that parted picks the first hard disk drive in the system.

You know, the hard disk that has all your system partitions and your home partitions on it? Accidentally working on that disk could be a bad idea. Notice here I used the select command to change the disk that I'm working on to the second hard disk in the system, which is /dev/sdb.

After you select the appropriate hard disk drive, you can create the new partition using the mkpart command. Then you have to specify a name for the partition. You have to specify the file system you want to use on the partition.

Which is interesting because if you create a partition with gdisk, you have to come in later after the fact and create a file system on it--which basically formats the partition so you can actually save information on it. Well, parted does that for you during the process of creating that new partition. Then you have to specify the starting point on the disk for the partition, and you do it in megabytes.

In this case, I am going to create a partition that starts at about the 11 GB point on the disk, so I specified a value of '11264', and then I have specified the ending point on the partition, again in megabytes. I am going to create a little 4 GB partition on the disk, so I am going to end at around 15 GB.

To view the partition that have been created on the disk, you can use the print command at the parted prompt. You can also use the other commands shown here to manage your partitions with parted.

To rename a partition, you use the name command followed by the name you want to enter. To move a partition, you use the move command, and you have to specify which partition you want to move and then where you want to move it to with the start and ending points.

If you want to resize a partition, use the resize command--by the way, this is really useful thing to be able to do--followed by the name of the partition you want to resize, and then its new start and end points. Or, if you just want to delete a partition, use the rm command followed by the name of the partition you want to remove.

Summary 10:42-10:52

That's it for this lesson. This lesson discussed how to manage GUID or GPT partitions on a Linux system. We first looked at how to do this using the gdisk utility. Then reviewed how to do this same thing using the parted utility.
