8.2.2 Managing GUID Partitions

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

Manage GUID Partitions 0:00-0:42

In this demonstration we're going to practice managing GUID partitions. GUID partitions cannot be managed with the old fdisk utility. Instead, we're going to look at using the gedit utility to manage GUID partitions and then we will look at using the parted utility to manage GUID partitions.

In order to do this, I do need to elevate my privileges on the system to root. Do that with the 'su' command. In this system I have two hard disks installed. sda is my system hard disk where my Linux operating system is installed; sdb is a new hard disk drive that doesn't have any partitions on it yet--it's blank. That's the one we want to work with.

'gdisk' Command 0:43-6:27

If you're already familiar with the fdisk utility for managing MBR partitions, then you're in luck because the gdisk utility works in much the same way as the fdisk utility. Just like with fdisk, we run 'gdisk' and then we specify the name of the device file we want to manage: '/dev/sdb'. When it does, it scans for partition table and this is a brand new hard disk drive, so there's nothing on it, and it reports that no partition tables were found.

The first thing we want to do is run the '?' command to view the help so we can see the list of all the commands that we can use. And you'll notice that they're very similar to those that are used by fdisk.

For example, to print the partition table, we can use the 'p' command. And we already know that there's no partitions on it, so let's go ahead and create a new partition.

To do that we use the n command just like with fdisk. We enter 'n', and now we have to specify the partition number that we want to create. And now here's where things get a little bit different from MBR partitions and fdisk.

Remember with MBR partitions, we can only create a maximum of four partitions per disk--either primary or extended partitions. Well, that's not the case with gdisk. With gdisk we can create up to 128 different partitions on a given disk. And there's no such thing as primary or extended partitions with GUID partitions. All partitions are just a partition.

Let's create partition number '1'. We have to specify the first sector where we want to start this partition. We'll press Enter to use to default of 2048. And now we can either specify the last sector that we want to use or just specify the size.

If you wanted to, you could use math to determine what the last sector should be. Notice up here that each sector is 512 bytes in size, so you can use that to determine how many sectors you would need to reach the desired size. In my case, I want to create one that is 10 GB in size.

This is a 20 GB hard drive, I want to use half of it for this first partition. But I'm really lazy so I'm not going to do the math, I'm just going to put a plus sign in, followed by the size, and notice that we have to specify whether we want to use kilobytes, megabytes, gigabytes, terabytes, and so on. We'll just use 'G' to specify 10 GB, and it automatically calculates my last sector for me.

The next thing we have to do is specify the type of file system we want to create. To do this we can enter 'l' to view a list of available codes. Notice here that to create a standard Linux partition you use code 8300, which is similar to the code 83 we use with MBR partitions to create a Linux file system.

However, if I wanted to say create a Linux swap partition, I would use code 8200. We want to just go ahead and use 8300 to create a standard Linux partition. Press Enter.

Now if we use the 'p' command we can view a list of the partitions and see the new partition that we created. It starts on sector 2048 and ends on sector 20973567 and is 10 GB in size and is a standard Linux partition.

Let's suppose that at some point I want to take this hard disk out of this system and then access at least some of the data on it, on a Windows system. Windows systems do not get along with Linux partitions at all, so we need to create a different type of partition to store that data.

We'll do 'n' again to create a new partition, this will be number '2'. We will use the default first sector that's available that will come right after the ending sector of this first partition, as you can see right here. And let's make it also 10 GB in size.

Well it didn't take the 10 GB. There must not quite be enough sectors to get exactly 10 GB, so we'll just go ahead and press Enter to accept the last ending sector that it's providing for us, which will basically just use the rest of the disk.

Now we have to specify what type of partition we want to create. Let's press 'l' again. And this time we want to use this hex code right here, 0700, Microsoft basic data.

By using this type of partition, the data we save on it can be readable on a Windows system. Let's enter '0700' for our partition type, print, and now we have two partitions defined on the system.

Just as with fdisk and MBR partitions, all the changes we've been making are actually only made in memory. They're not actually committed to disk. gdisk provides you with the flexibility to play around with these partitions, playing with them until you finally get them the way you want.

Once you do, then you use the 'w' command to actually write them to the hard disk drive. And it warns us, "By the way, this is going to overwrite anything that may be on the disk. Are you sure you want to do this?" If there were data in an existing partition on this disk that you wanted to save, you should back it up before you actually write these partitions to disk.

We don't have anything on this one, so we're good to go. I'll press Enter, I press 'y', and then press Enter to write the changes. And it was successful.

Let's print the partition table real quick. And we can see that we know have these two partitions on the hard disk drive. There may be times when you need to delete partitions, and with gdisk it's done in the exact same way as it is with fdisk.

If you look here we use the 'd' command to delete a partition. Let's delete partition number 2, print, and let's delete partition number 1. Print again, they're gone. Let's do a 'w' to write the changes to disk. It warns us that we are about to destroy data. We know. We'll write it out to disk. And now our hard disk drive has been cleaned off. That's how you use gedit to create GUID partitions.

'parted' Command 6:28-9:53

There's another utility you can use as well called parted. Like gdisk, parted can be used to add, delete, edit the partitions on your disk. However, there's one caveat that you have to be familiar with when working with parted, which by the way stands for partition editor.

And that is the fact that unlike gdisk and fdisk for that matter, the parted utility writes the partition changes that you specify immediately to the disk. With gdisk, we can play around with the partitions for a while and get them the way we like them and then write them to disk. That's not the case with 'parted'. It writes it as you do it.

To use parted, we enter the 'parted' command in the command prompt. Here's another gotcha that you have to be aware of with parted. Notice that by default it's picked sda to work on.

Well that's my system hard drive, that's where my Linux system is installed, I do not want to mess with the partitions on that disk. What we have to do is tell parted to not use sda but instead to 'select /dev/sdb' instead. Now we're working on sdb.

Now that we've selected the appropriate hard disk, we can create a new partition using the 'mkpart' command. I'll enter that at the prompt. We have to give the partition a name, let's call it 'DATA'. Here's where things are very different between parted and gdisk.

Remember with gdisk all we do is create partitions; we don't create file systems on those partitions we have to do that later on. parted on the other hand creates the partition and allows us to define a file system on that partition all at once. And now since we're partitioning and formatting the partition all at same time, so we can save data on it.

We're being prompted to specify what type of file system we want to create on that new partition. Let's go ahead and use 'ext4'. Now we have to specify the beginning of the partition on the disk. And you do this in a different way than you do with gdisk.

With parted you have to specify the starting point on the disk in terms of megabytes. What I essentially want to do is start at '0' MB on the disk to start at the beginning. And then I have to specify where I want the partition to end.

For example, if I wanted to create a 10 GB partition, well 10 GB is '10,240' MB. So that's where I'll set my ending point for the partition.

And it's warning me that the resulting partition is not properly aligned for best performance. We better cancel that. Let's do a cancel. I think we actually have to start the partition at the 2 MB point. I forgot about that, so let's do it again.

Let's do a 'mkpart', 'DATA', we'll use 'ext4'. Let's start at the '2' MB point on the disk. Let's end it at '10,240' MB, or 10 GB, press Enter. And it liked it that time. At this point, the partition has actually been written to disk.

We can actually use the 'print' command to view it. And here you see partition number 1: here's where it starts, here's where it ends, total size and it uses the ext4 file system. In order to get out of parted we just type 'quit'.

Summary 9:54-10:04

That's it for this demonstration. In this demo we talked about how to manage GUID partitions. First we looked at managing GUID partitions using the gdisk command, and then we ended this demonstration by looking at using the parted command.
