4.2.4 GRUB2 Bootloader

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

GRUB2 Bootloader 0:00-0:30

Let's spend some time talking about the GRUB bootloader. GRUB is very powerful. In fact, it's the most widely used Linux bootloader. Nearly all modern Linux distributions will use the GRUB bootloader by default. The acronym GRUB stands for Grand Unified Bootloader. The job of the bootloader is to load the Linux operating system kernel from your system's hard drive when the system powers on. In fact, it can be used to boot any other operating system's kernel from the hard drive as well, such as Windows.

Different Versions of GRUB 0:31-1:13

Now, be aware that there are two different versions of GRUB that you need to be familiar with.

The first one, the older one, is the GRUB legacy bootloader. The GRUB legacy bootloader was very popular, and it was very widely used for a number of years.

However, recently, there's been a steady shift away from GRUB legacy towards a newer version of GRUB called GRUB2. In fact, most Linux distributions that are available today will use GRUB2 instead of GRUB legacy.

If you're not sure which version of GRUB your particular distribution is using, you can use either of these two commands that you see here, and they will report the GRUB version.

You can use 'grub-install -V', or you can enter 'grub2-install -V'.

GRUB Configuration Files 1:14-2:00

With that in mind, let's take a look at the configuration files that are used by GRUB. Understand that GRUB will come pre-configured to boot your operating system's kernel, but you don't have to leave it in its default configuration; you can change it. In order to do that, you need to be familiar with these three files. First of all, the /boot/grub/grub.cfg file; all of the files that exist inside this directory right here, /etc/grub.d; and then, finally, the /etc/default/grub file.

Before we move on, I do need to point out that, depending upon your distribution, you may find that your grub.cfg file is not located in /boot/grub. You may, on some distributions, find it located in /boot/grub2/grub.cfg. If you can't find it in one of these directories, try the other one, and you will find it there.

GRUB2 Configuration Update Commands 2:01-3:21

Here's the key thing that you need to understand about GRUB2: that is the fact that you should not directly edit the grub.cfg file. What you should do is, instead, make the appropriate changes to one of the other configuration files that we'll talk about in just a minute. And then, once you've done that, you then need to run one of these two commands. When you run these commands, the grub.cfg file is automatically updated with the appropriate information, and then it will be read by the bootloader when the system boots and, therefore, configure the boot process. Some distributions, such as Ubuntu, will use the update-grub command to update the grub.cfg file. Other distributions, such as OpenSUSE, Fedora, and so on, will use the grub2-mkconfig command. Notice, when you run update-grub, say, on an Ubuntu system, you don't have to specify the name of the GRUB configuration file in the command. But if you're using OpenSUSE, Fedora, or some other distribution that uses grub2-mkconfig, then you use the -o option, right here, to specify the output file name, which should be usually /boot/grub2/grub.cfg. And, again, be aware of some distributions that may just be /boot/grub/grub.cfg. Make sure you know the name of your GRUB configuration file before you run this command; otherwise, your changes won't be implemented because the GRUB bootloader will be reading the wrong file.

GRUB2 Menu Configuration 3:22-5:06

Now, your actual GRUB2 menu configuration is stored in the configuration files located in the /etc/grub.d directory. These aren't your typical configuration files, where we have directive and we set that directory to a particular value. Instead, all of these files that you see here are script files. So, if you make changes to one of these files and then run the update-grub or the grub2-mkconfig command, these scripts will be run, which will then be used to automatically generate the grub.cfg file with the new updated settings that you specified.

I want you to pay very close attention to the naming convention that is used for the files in the /etc/grub.d directory. Typically, what you'll see is that they all begin with a number instead of a character. For example, 00_header, 10_linux, 20_linux_xen, 30_os-prober, 40_custom, and so on. This naming convention is important because the placement of the menu items within the grub.cfg file is determined by the numeric order of the script file names in /etc/grub.d. Basically, the lowest number files are executed first when you run the update-grub or the grub2-mkconfig command, and then the higher number files are run afterwards. In this example, 00_header would be run first, 10_linux would be run second, and so on. If there happen to be any script files in this directory that do not use a number for the filename, they will not be run until after all the script files that do use numbers at the beginning of their filenames are run. This then influences the order in which the information, all these script files, appear in your /boot/grub2/grub.cfg file.

Commonly Used grub.d Script Files 5:07-5:28

With this in mind, let's take a look at some of the commonly used script files within grub.d. The files that you see here are those that were used on this particular distribution; it was a Fedora system. The actual files that will be used will vary from distribution to distribution; it'll be customized by that particular vendor. We're going to look at those that are usually seen across most distributions.

00_header 5:29-5:44

The first one you need to be familiar with is 00_header.

This file configures various parameters for the GRUB2 boot menu, such as the initial appearance of the boot menu screen, the graphics mode, the default selection, the timeout values, and so on.

10_linux 5:45-6:10

The next file you need to be familiar with is 10_linux.

This file identifies each of the Linux kernels that have been installed on your system because you can actually have multiple Linux kernels installed. If you do, then the 10_linux script will actually search through your boot folder and locate all of the installed Linux kernels. For each one that it finds, it's going to create a separate menu entry in your GRUB2 boot menu.

30_os-prober 6:11-6:46

The next one that you need to be familiar with is 30_os-prober.

This script searches for any other operating systems that may be installed on your system. If you have just Linux installed, then this script will locate your Linux kernel and create a menu entry for it. However, if you have a dual boot system, and you have Linux installed as well as Windows in separate partitions on the same system, then 30_os-prober will locate these two operating systems that have been installed and will automatically create a GRUB menu entry for each one. That allows you to boot either Linux or Windows from your initial boot screen.

40_custom 6:47-8:20

The last one you need to be familiar with is 40_custom. This file is intended for you to use as a means to customize the behavior of the GRUB menu file. It's essentially just a template that you can use to add custom entries that'll be inserted into your grub.cfg file. Notice that because this file begins with 40, it will be one of the last files that gets run when you update your GRUB configuration.

Now, I'll be straight with you. Most of the time, you only use the script files in grub.d to add or remove menu items from your GRUB boot menu, say, if you have multiple operating systems installed.

If you need to configure other basic parameters of your GRUB boot menu, instead of editing one of the script files, usually what we do is just go in and edit the /etc/default/grub file.

Now, could you make these same changes in the script file, grub.d? Yeah. But the script files are just a lot harder to work with because they're script files instead of traditional configuration files.

If you need to just make basic changes, such as the way the GRUB menu looks, maybe timeout value, what the default selection is, and so on, it's actually just easier to come into /etc/default/grub and make the changes there, because when you run your update command, this file will be read, as well as the script files to customize your GRUB boot menu. You can just open up this file in a text editor, make whatever changes you need to, and those changes will get incorporated into grub.cfg when you run update-grub or grub2-mkconfig.

GRUB_DEFAULT 8:21-9:04

There are several key settings within the grub.cfg file that you need to be aware of.

The first one is GRUB_DEFAULT. This parameter sets the default menu entry. For example, if you were to set GRUB_DEFAULT to a value of 0, then the very first menu entry would become the default menu entry. And if you have a timer set and no selection is made manually by the end user, then that first menu entry would be automatically selected. In fact, that's the default configuration for most distributions when you install them.

You're not stuck with this. If you want to have, say, the second menu entry to be the default, then you would edit this file and change GRUB_DEFAULT to a value of 1, which would boot the second menu entry.

GRUB_SAVED_DEFAULT 9:05-9:45

The next value you need to be familiar with in this file is GRUB_SAVED_DEFAULT.

If you set this parameter to a value of true, then what GRUB is going to do is check and see what the last menu item that was selected the last time the bootloader ran, and it will use that as the default selection the next time.

With GRUB_DEFAULT, we hard select, if you will, which menu item is the default. If we use GRUB_SAVEDEFAULT, on the other hand, then the default menu entry will vary. As you can see, these two parameters could potentially conflict with each other. As a general rule of thumb, you should use either GRUB_DEFAULT or GRUB_SAVED_DEFAULT, not both.

GRUB_HIDDEN_TIMEOUT 9:46-10:53

You should also be familiar with the GRUB_HIDDEN_TIMEOUT parameter.

This parameter can be used to cause the boot process to pause and then display a blank screen (or a splash image, if you decide to go that direction) for a specified number of seconds. Then, at the end of that timeout period that you specify, the system will actually boot.

This parameter determines how long that timeout period will be--how long that blank screen will be displayed. It's important to note that while that screen is blank during this timeout period, the user can, if they want to, press the Shift key to display the menu.

It's important to note that this parameter is actually optional. In fact, I don't really like it, so I don't use it. But if you do, then you do have to use this value in conjunction with it.

If the GRUB_HIDDEN_TIMEOUT_QUIET value is set to true, then no countdown timer will be displayed. If you set it to false, then a countdown timer will be displayed on the blank screen for the duration of the period specified in the GRUB_HIDDEN_TIMEOUT value.

GRUB_TIMEOUT 10:54-12:06

As I said, I don't actually like these two parameters very much. What I do use, though, is this one right here, GRUB_TIMEOUT.

This parameter specifies how long, in seconds, the user has to make a selection from the GRUB menu before the default menu item will be automatically selected. Be aware that if you set this parameter to a value of -1, it will actually pause the GRUB menu to not automatically select any selection at all. Basically, it will just boot up to the GRUB menu, and then it will stay there until the end user makes a selection manually from the menu.

Some folks like to do this, but I actually don't. If I'm powering on, say, my data center, and I'm pushing power buttons on server systems, I don't want to have to go through each one of them and manually select a menu item to get the system up and running. I want to push the power button, and I want them to automatically come up because usually my server systems are what we call headless--they don't have a monitor directly attached, or they may be sharing one monitor. I don't want to have to cycle through my KVM switch trying to find each different system and making sure that it's booting properly; I just want to push the power button. But there may be situations where you don't want that to happen. You want the system to pause and wait for the end user to manually select which kernel you want the bootloader to load.

GRUB_CMDLINE_LINUX 12:07-12:31

The next parameter is GRUB_CMDLINE_LINUX. This parameter is used to pass options to the kernel. This is really useful. Back in the old days, with the GRUB legacy bootloader, you did this by adding options to the end of the kernel line itself, and that was a really handy thing to do. It's not quite the same with GRUB2. You can still accomplish the same thing, though, by adding those parameters to GRUB_CMDLINE_LINUX.

GRUB_GFXMODE 12:32-13:38

The next one you need to be familiar with is GRUB_GFXMODE.

This parameter sets the resolution of the graphical menu displayed by GRUB2, and you can also add a color depth as well. For example, if I wanted to set the GRUB menu to use a screen resolution of 1440 X 900 and to use 24-bit color depth, I would set this parameter to a value of 1440x900x24.

You can also specify multiple resolutions if you need to. Basically, what you should do is put your preferred screen resolution first, then add a comma, then add your second preferred resolution and a comma, and then your third resolution, and so on.

When you do, GRUB will try the first screen resolution first. If that resolution isn't supported by the video display, it'll move on to the second one, and so on until it hits one that works.

If this parameter is actually commented out or isn't listed at all, then GRUB2 will just use the default graphic mode setting found in the /etc/grub.d/00_header file.

GRUB_DISABLE_OSPROBER 13:39-14:26

The last parameter we're going to look at here is GRUB_DISABLE_OSPROBER.

If you set this parameter to a value of true, then this will actually disable the osprober check. Remember, the osprober check looks for partitions on the hard drive that have other operating systems installed, such as a Windows installation, or maybe another Linux distribution. If you want GRUB to look for those, you want to make sure that this parameter is set to false or is not even included. If you want to disable that functionality because you don't want menu items added for those other operating systems, then set this parameter to a value of true.

The important thing that you must remember is that any time you make any change at all to this configuration file, you have to run either 'update-grub' or 'grub2-mkconfig' in order for the change to be applied. If you don't, it won't be applied.

Summary 14:27-14:38

That's it for this lesson. In this lesson, we introduced you to the GRUB2 bootloader. We reviewed the configuration files that are used to customize the bootloader, and then we ended this lesson by looking at several key configuration settings that you may want to change to customize the way the GRUB2 bootloader works.